UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
CURSO DE ENGENHARIA MECÂNICA

FELIPE PADILHA CRISCUOLI
BEATRIZ LEAL MIRANDA

CONTROLE PID DE MOTOR DC DE BAIXA POTÊNCIA IMPLEMENTADO EM UM MICROCONTROLADOR PIC

VITÓRIA
2009

FELIPE PADILHA CRISCUOLI
BEATRIZ LEAL MIRANDA

CONTROLE PID DE MOTOR DC DE BAIXA POTÊNCIA IMPLEMENTADO EM UM MICROCONTROLADOR PIC

Projeto de Graduação apresentado ao Departamento de Engenharia Mecânica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Mecânico, sob orientação do Prof. Dr. Rafael Luís Teixeira e co-orientação do Eng. Salim Suhet Mussi.

VITÓRIA
2009

MIRANDA, Beatriz Leal; CRISCUOLI, Felipe Padilha.

Controle PID de motor DC de baixa potência implementado em um microcontrolador PIC. / Beatriz Leal Miranda; Felipe Padilha Criscuoli – 2009.

116f.

Orientador: Rafael Luís Teixeira

Co-orientador: Salim Suhet Mussi

Projeto de Graduação – Universidade Federal do Espírito Santo, Centro Tecnológico, Departamento de Engenharia Mecânica.

1. Controle. 2. Microcontrolador. 3. PIC. 4. PID. 5. PWM. 6. Controle de velocidade. 7. Motor DC.

I. MIRANDA, Beatriz Leal. II. Criscuoli, Felipe Padilha. III. Universidade Federal Do Espírito Santo, Centro Tecnológico, Departamento de Engenharia Mecânica. IV. Controle PID de motor DC de baixa potência implementado em um microcontrolador PIC.

FELIPE PADILHA CRISCUOLI
BEATRIZ LEAL MIRANDA

CONTROLE PID DE MOTOR DC DE BAIXA POTÊNCIA IMPLEMENTADO EM UM MICROCONTROLADOR PIC

Projeto de Graduação apresentado ao Departamento de Engenharia Mecânica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Mecânico.

Aprovado em ___ de _____ de 2009.

COMISSÃO EXAMINADORA:

_____

Prof. Dr. Rafael Luís Teixeira
Orientador

_____

Eng. Salim Suhet Mussi
Co-orientador

_____

Prof. Dr. Marcos Aurélio Scopel Simões
Examinador

_____

Eng. João Victor Santos Bissoli
Examinador

Aos nossos pais e irmãos, que compensam todos os dias o esforço e trabalho que tivemos com demonstrações de afeto e apoio incondicional. A estes que são a nossa base, referência e orgulho maior.

“Muitas das grandes realizações do mundo foram feitas por homens cansados e desanimados que continuaram trabalhando”

Autor desconhecido

AGRADECIMENTOS

Agradecemos um ao outro e também a todos que tornaram esse projeto possível.

Ao nosso orientador Rafael por difundir seu amplo conhecimento na área de sistemas de controle e por colaborar com suas idéias.

Ao amigo e co-orientador Salim, por nos guiar nesse processo, com seu conhecimento e paciência.

Ao Hugo Tanzarella, por disponibilizar os equipamentos e tempo na viabilização desse projeto e dar suporte nos testes de bancada realizados no LECO.

À McNose Vintage Tube Amplifiers por disponibilizar componentes e know-how em eletrônica.

Aos nossos colegas do LabDin: Matheus, Elton, Túlio, Daniel e Fabrício e ao Tiago, que além de dividirem conhecimento e estarem sempre prontos para ajudar, trouxeram diversão à elaboração desse trabalho e se tornaram família quando o LabDin se tornou nossa casa.

À Deus por nos dar força a ânimo, principalmente nos momentos de dificuldades e contratempos.

E finalmente àqueles que embora não tenham contribuído de maneira direta nesse trabalho, são as pessoas sem as quais jamais chegaríamos até aqui: nossos pais e irmãos. Obrigado pela dedicação e carinho ao longo de toda a nossa jornada.

RESUMO

O presente trabalho apresenta uma metodologia de desenvolvimento e construção de um módulo de controle PID através de modulação por largura de pulso (PWM) para motores de baixa potência, para tanto é feita uma revisão de conceitos de motores de corrente contínua, sensores para medição de velocidade, microcontroladores, programação e controle de sistemas, com a visão de integrar conhecimento de diferentes áreas da engenharia.

O módulo proposto possibilita a análise, em malha aberta ou fechada, de diferentes tipos de sinais de entrada no sistema e também a modificação dos parâmetros de controle em tempo real.

Palavras-chave: Controle, Microcontrolador, PIC, PID, PWM, Controle de velocidade, motor DC

ABSTRACT

The current project presents a methodology for developing and building a PID control module using Pulse Width Modulation (PWM) for low power motors. A review of basic concepts concerning direct current engines, sensors for speed measurement, microcontrollers, programming and system control is also presented, mixing knowledge from various engineering areas.

The proposed module allows open-loop and feedback control analysis for different kinds of signal inputs and also a real-time adjustment of the control parameters.

Key-words: Control, Microcontroller, PIC, PID, PWM, speed control, DC motor

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em um mundo onde a disponibilidade em larga escala de microprocessadores e microcontroladores com custo cada vez menor e desempenho cada vez maior é um fato, a integração entre sistemas mecânicos, elétricos e processos computacionais para controle desses sistemas é crescente e com isso os conhecimentos das áreas tradicionais da engenharia – mecânica, elétrica e ciência da computação – precisam se misturar para que haja total entendimento dos sistemas em questão e se desenvolva a capacidade de projetar novos sistemas desse tipo ou mesmo propor melhorias àqueles já existentes.

Cetinkunt (2004) define mecatrônica como a área sendo a interseção entre as áreas tradicionais da engenharia supracitadas. Ainda segundo Cetinkunt (2004), "[...] ela representa o estágio atual da evolução dos campos da engenharia que lidam com o projeto de sistemas eletromecânicos controlados". A Figura 1.1 mostra a caracterização da mecatrônica de acordo com o autor.

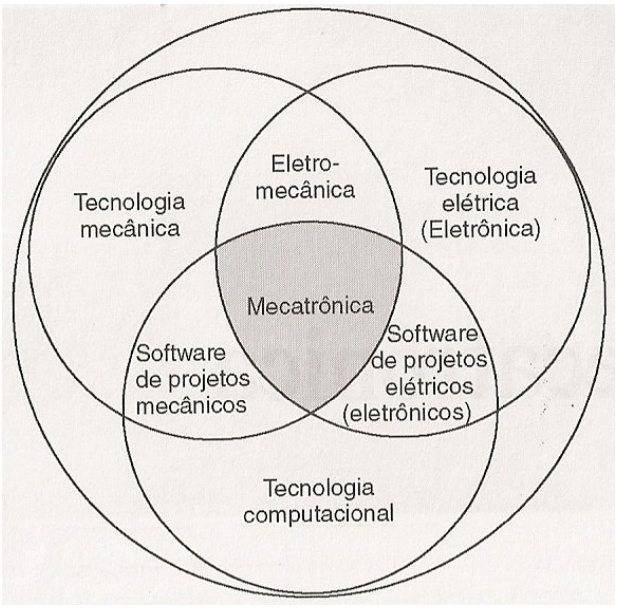


Figura 1.1 – Mecatrônica
Fonte: Cetinkunt (2004)

O número de opções que se abrem quando tornamos o conhecimento multidisciplinar é imenso e não acompanhar a evolução da tecnologia significa tornar esse conhecimento obsoleto.

Nesse trabalho serão abordados alguns conceitos básicos sobre motores, sensores de velocidade, microcontroladores e linguagem de programação “C”, tipos de controle e suas características, além do projeto de um módulo PID.

1.1 OBJETIVO GERAL

O presente trabalho tem como objetivo mostrar o desenvolvimento e construção de um módulo de controle PID para motores de baixa potência através de modulação por largura de pulso (PMW, do inglês *Pulse-Width Modulation)* e utilizando o microcontrolador PIC®16F877A. Com esse módulo será possível variar os valores dos ganhos proporcional, integral e derivativo (Kp, Ki e Kd, respectivamente) e obter um sinal de controle adequado através do ajuste dos mesmos.

1.2 OBJETIVOS ESPECÍFICOS

Agregar conhecimentos em eletrônica e na utilização de microcontrolodaores aos alunos e professores do LabDin (Laboratório de Dinâmica) e ao curso de engenharia mecânica.

Desenvolver uma ferramenta útil aos professores do departamento, da área de automação e controle, que sirva de base para futuras pesquisas sobre controle de motores CC.

Aplicar os conhecimentos gerados durante a etapa de pesquisas, corroborando com o caráter de pesquisa e aplicação que se espera de um projeto de graduação de engenharia.

1.3 JUSTIFICATIVA PARA ESCOLHA DO TEMA

A necessidade de integrar o conhecimento, como foi destacada anteriormente, foi a principal motivação para a escolha desse tema. O intuito foi o de desenvolver um projeto que gerasse conhecimento pouco explorado no curso de engenharia mecânica, na área de mecatrônica, servindo como ponto de partida para a expansão dessa área.

2 MOTORES

Motor elétrico é uma máquina destinada a transformar energia elétrica em mecânica. É o mais usado de todos os tipos de motores, pois combina as vantagens da energia elétrica (baixo custo, facilidade de transporte, limpeza e simplicidade de comando) com uma construção simples, custo reduzido, grande versatilidade de adaptação às cargas dos mais diversos tipos e melhores rendimentos.

Os motores elétricos são divididos em dois grandes grupos:

- Motores de corrente contínua;
- Motores de corrente alternada;

Atualmente, o desenvolvimento das técnicas de acionamentos de corrente alternada (CA) e a viabilidade econômica têm favorecido a substituição dos motores de corrente contínua (CC) pelos motores de indução acionados por inversores de freqüência. Apesar disso, devido às suas características e vantagens, que serão analisadas adiante, o motor CC ainda se mostra a melhor opção em aplicações como: Máquinas de Papel, laminadores, máquinas de impressão (Honda, 2006).

No trabalho será abordada apenas a utilização de motores DC (ou corrente contínua) devido ao fato desse ser largamente utilizado em situações de controle, objetivo do estudo.

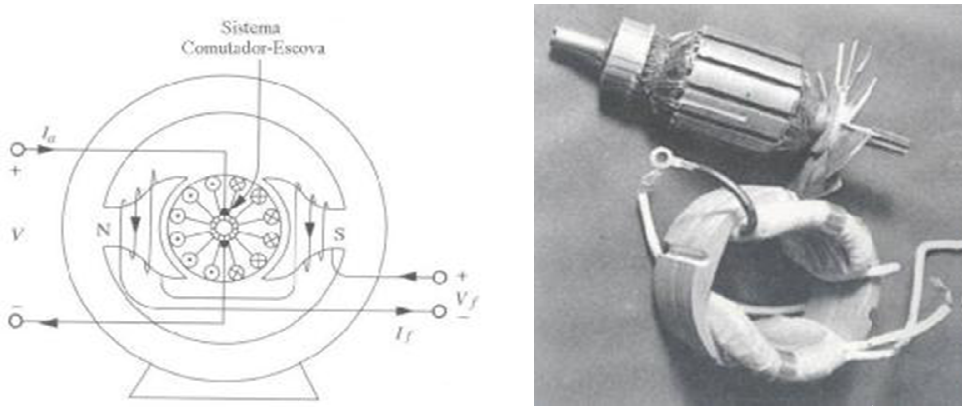
2.1 ASPECTOS CONSTRUTIVOS

O motor de corrente contínua é composto de duas estruturas magnéticas:

· Estator (enrolamento de campo ou ímã permanente);

· Rotor (enrolamento de armadura).

O estator é composto de uma estrutura ferromagnética com pólos salientes aos quais são enroladas as bobinas que formam o campo. A Figura 2.1 mostra o desenho de um motor CC de 2 pólos com enrolamento de campo.



(a) (b)

Figura 2.1 - (a) Desenho e (b) foto de um motor CC de 2 pólos com enrolamento de campo
Fonte: Honda, F (2006)

O rotor é um eletroímã constituído de um núcleo de ferro com enrolamentos em sua superfície que são alimentados por um sistema mecânico de comutação. Esse sistema é formado por um comutador, que possui uma superfície cilíndrica com diversas lâminas às quais são conectados os enrolamentos do rotor; e por escovas fixas, que são ligadas aos terminais de alimentação. O propósito do comutador é o de inverter a corrente na fase de rotação apropriada para que seja mantido o conjugado (momento) sempre na mesma direção.

Os enrolamentos do rotor compreendem bobinas de n espiras Os dois lados de cada enrolamento são inseridos de modo que quando os condutores de um lado estão sob o pólo norte, os condutores do outro devem estar sob o pólo sul.

2.2 PRINCÍPIO DE FUNCIONAMENTO

O funcionamento de um motor CC de dois pólos é descrito, de maneira simplificada pela Figura 2.2, em quatro situações (a, b, c, e d) que representam o giro do rotor e o efeito do campo magnético gerado pelas bobinas.

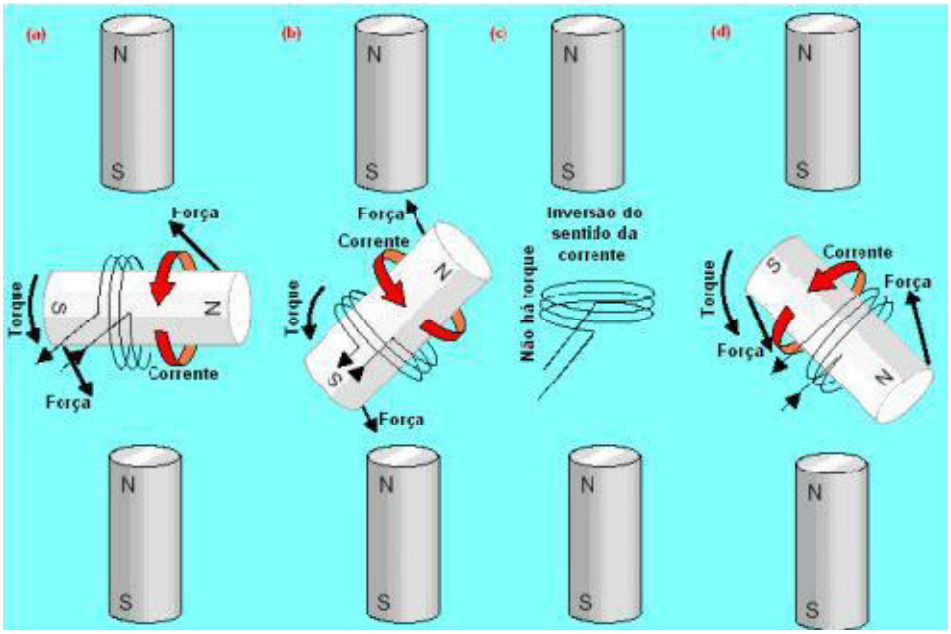


Figura 2.2 - Funcionamento do motor CC de dois pólos

Fonte: Honda, F (2006)

A Figura 2.2 representa um desenho esquemático simples de um motor onde o estator é constituído por ímãs permanentes e o rotor é uma bobina de fio de cobre esmaltado por onde circula uma corrente elétrica. Uma vez que as correntes elétricas produzem campos magnéticos, essa bobina se comporta como um ímã permanente, com seus pólos N (norte) e S (sul) como mostrados na figura.

Na situação ilustrada em (**a**) os pólos opostos se atraem, a bobina experimenta um torque que age com o intuito de girar a bobina no sentido anti-horário. A bobina sofre aceleração angular e continua seu giro para a esquerda, como se ilustra em (**b**).

Esse torque continua até que os pólos da bobina alcancem os pólos opostos dos ímãs fixos (estator). Nessa situação (**c**) – a bobina girou de 90° – não há torque algum, pois os braços de alavanca são nulos; o rotor está em equilíbrio estável (força resultante nula e torque resultante nulo). Esse é o instante adequado para inverter o sentido da corrente na bobina.

Agora os pólos de mesmo nome estão muito próximos e a força de repulsão é intensa. Devido à inércia do rotor e como a bobina já apresenta um momento angular, ela continua girando no sentido anti-horário e o novo torque (agora propiciado por forças de repulsão), como em (**d**), colabora para a manutenção e aceleração do movimento de rotação (HONDA, F. 2007).

Apesar de não estar esquematizado na figura, o movimento continua similar quando o rotor chega a 270° e, após chegar à 360° e o ciclo se repete.

Essas atrações e repulsões bem coordenadas é que fazem o rotor girar. A inversão do sentido da corrente (comutação), no momento oportuno, é condição indispensável para a manutenção dos torques ”favoráveis”, os quais garantem o funcionamento dos motores.

Durante a comutação, a bobina é momentaneamente curto-circuitada pelas escovas, o que ajuda a liberar energia armazenada, antes de a corrente fluir no sentido oposto. Porém, como essa inversão de corrente não é instantânea, uma força eletromotriz é induzida na espira, segundo a equação 2.1.

$$Fem = \left(-L_a \frac{di_a}{dt}\right) \qquad 2.1$$

O que origina uma corrente de curto-circuito que circula no coletor, nas espiras e nas escovas.

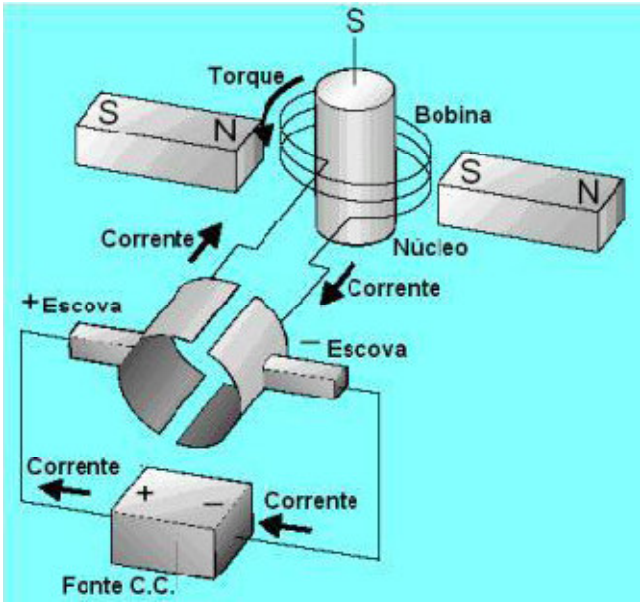


Figura 2.3 - Comutador de escovas
Fonte: Honda, F (2007)

Em sua forma mais simples, o comutador apresenta duas placas de cobre encurvadas e fixadas (isoladamente) no eixo do rotor; os terminais do enrolamento da bobina são soldados nessas placas. A corrente elétrica “chega” por uma das escovas (+), “entra” pela placa do comutador, “passa” pela bobina do rotor, “sai” pela outra placa do comutador e “retorna” à fonte pela outra escova (-). Nessa etapa o rotor realiza sua primeira meia-volta. Nessa meia-volta, as placas do comutador trocam seus contatos com as escovas e a corrente inverte seu sentido de percurso na bobina do rotor. E o motor CC continua girando, sempre com o mesmo sentido de rotação (HONDA, F. 2007).

2.3 CONTROLE DE VELOCIDADE NOS MOTORES CC

O modelo do circuito elétrico do motor CC é ilustrado na Figura 2.4

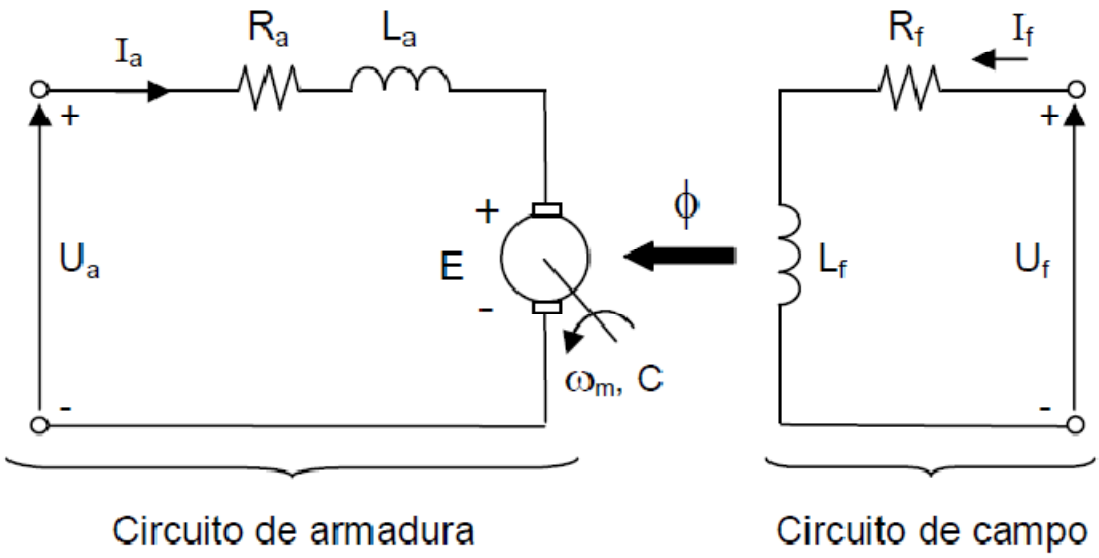


Figura 2.4 - Modelo do Circuito elétrico de um motor CC

Fonte: Honda, F (2006)

A Lei de Kirchhoff aplicada ao circuito de armadura resulta na equação:

$$U_a = R_a * I_a + E \qquad 2.2$$

Onde:

Ua = Tensão de armadura

Ra = Resistência da armadura

Ia = Corrente de armadura

E = Força Eletromotriz induzida ou Força Contra-Eletromotriz da armadura

Pela Lei da Indução de Faraday, a força eletromotriz induzida é proporcional ao fluxo e à rotação, ou seja:

$$E = k_1 * \emptyset * n \qquad 2.3$$

Combinando as eq. 2.2 e 2.3 a expressão para a velocidade do motor CC é dada por:

$$n = k_1 \frac{Ua - Ra * Ia}{\emptyset} \qquad 2.4$$

Onde:

n = velocidade de rotação

k1 = constante que depende do tamanho do rotor, do número de pólos do rotor, e como esses pólos são interconectados.

f = fluxo no entreferro

Admitindo-se que a queda de tensão na armadura é muito pequena:

$$R_a * I_a \cong 0 \qquad 2.5$$

A equação 2.2 se reduz a:

$$n = k_1 \frac{U_a}{\emptyset} \qquad 2.6$$

Portanto, a velocidade de giro de um motor é diretamente proporcional à tensão de armadura, e inversamente proporcional ao fluxo no entreferro.

A partir dessa equação, pode-se concluir que o controle da velocidade, até a velocidade nominal, é feito através da variação da tensão de armadura do motor, mantendo-se o fluxo constante (HONDA, F. 2007).

Velocidades superiores à nominal podem ser conseguidas pela diminuição do fluxo, mantendo-se a tensão de armadura constante, entretanto não são o objetivo de estudo do trabalho e, portanto, serão desconsideradas.

3 MEDIÇÃO DE VELOCIDADE

Sistemas de controle são usados para estabelecer alguma relação funcional entre grandezas de entrada e de saída. Para tanto, necessita-se medir a grandeza de saída (variável a controlar), ou seja, para controlar uma grandeza, obviamente, é necessário que se meça a mesma. Essa medição determina a diferença entre o valor da variável de saída (real) e o valor desejável, para que se aplique uma correção (sinal atuante) capaz de reduzir e eliminar essa diferença.

Isso só se torna possível com a utilização de transdutores, que consistem, segundo Pedro A. (2004) em dispositivos capazes de transformar um tipo de sinal em outro para permitir o controle de processos físicos, ou realizar medições.

3.1 ENCODERS

Os encoders são dispositivos transdutores de movimento. Eles convertem o movimento, linear ou angular, em um sinal elétrico que pode ser adquirido e transformado em outras informações, como velocidade ou posição.

Um encoder consiste em um disco perfurado, ou máscara, que, dependendo da posição, permite ou não a passagem de luz, de um emissor contínuo de luz posicionado em um dos lados desse disco e de um receptor, posicionado do outro lado do disco, que com o auxílio de um sistema eletrônico, gera pulsos.

O sinal pode ser usado para determinar a velocidade ou a posição, partindo-se de uma referência e contando o número de pulsos gerados. O número de furos no disco determina a precisão do sensor. Um disco com 100 furos, por exemplo, irá gerar 1 pulso a cada 3,6° e um disco de 360 furos irá gerar 1 pulso a cada 1°, de forma que quanto maior o número de rasgos maior será a precisão. É ainda importante destacar que o encoder produz diretamente uma

saída digital, eliminando a necessidade de um conversor analógico-digital. A Figura 3.1 mostra o esquema de um encoder rotativo.

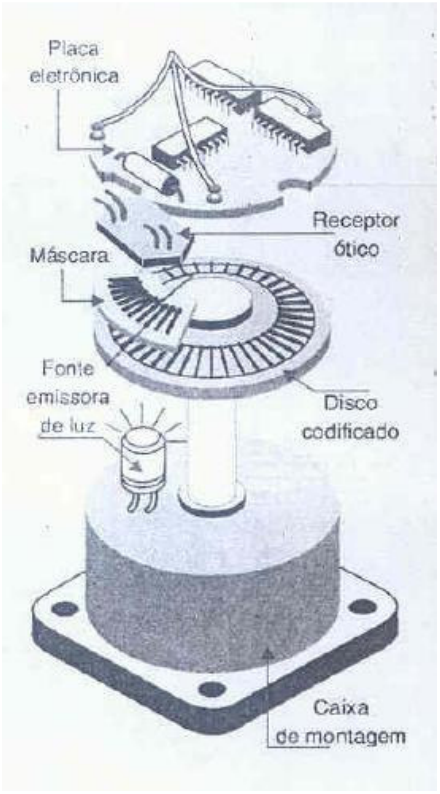


Figura 3.1 - Esquema de um encoder rotativo
Fonte: http://www.ucs.br/ccet/demc/vjbrusam/inst/enc1.pdf

De acordo com Kilian (2000), os encoders podem ser de dois tipos: incremental ou absoluto, dependendo de sua máscara.

3.1.1 Encoder incremental

Possui apenas uma trilha de furos, igualmente espaçados. A cada momento que o disco interrompe a passagem de luz, um pulso é gerado e com isso, a posição é medida a partir da contagem de quantas vezes a passagem de luz foi interrompida, partindo de um referencial pré-estabelecido. Já a velocidade é medida pela freqüência dos pulsos. A Figura 3.2 mostra o disco de um encoder incremental.

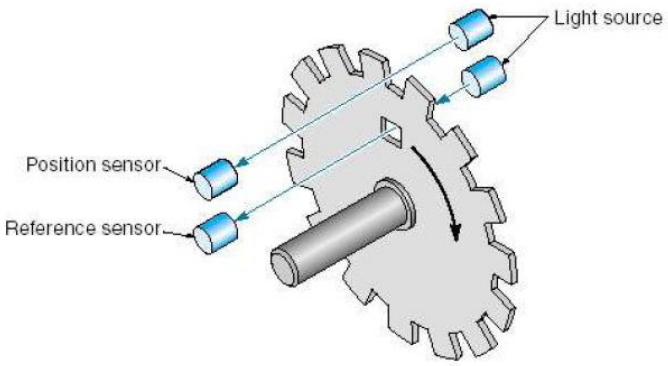


Figura 3.2 - Disco de um encoder incremental comum
Fonte: Kilian (2000)

Esse tipo de encoder é limitado, pois não consegue detectar qual é o sentido da rotação do eixo, mas o mesmo disco pode ser utilizado com dois fotosensores e um pequeno espaço entre eles de forma que a defasagem de um sinal em relação ao outro permita definir o sentido de giro. Nesse caso, teríamos a montagem mostrada na Figura 3.3 (a), para a qual o sinal de saída é mostrado na Figura 3.3(b).

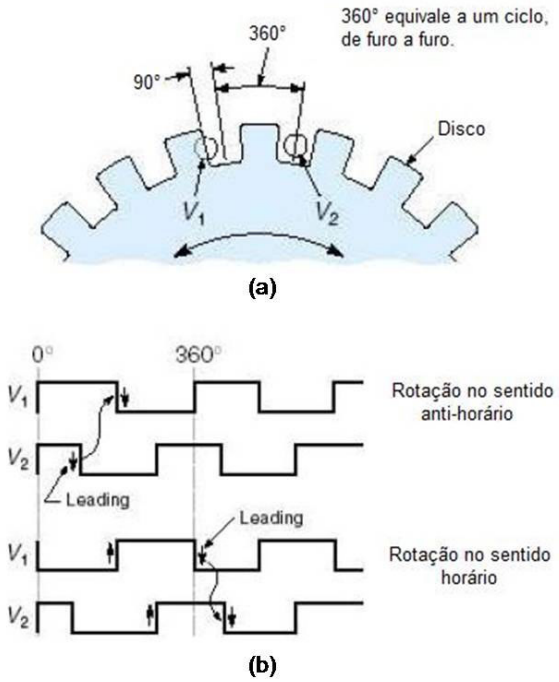


Figura 3.3 - (a) Esquema de encoder incremental com dois fotosensores e (b) seu sinal de saída

Fonte: Adaptado de Kilian (2000)

É muito comum o uso desses encoders em mouses e impressoras pelo fato do seu custo ser mais baixo do que o encoder absoluto, que será apresentado a seguir. Exatamente pela facilidade de se conseguir um encoder incremental, ele foi, à princípio, escolhido para ser o sensor utilizado na aquisição dos dados de saída do motor nesse projeto. A leitura desse sinal permite o controle do motor em malha fechada.

3.1.2 Encoder absoluto

No encoder absoluto, o elemento básico é um disco de vidro estampado com um padrão de trilhas concêntricas ou mesmo um disco com furos, desde que cada posição do disco tenha padrão único. Os feixes de luz atravessam cada trilha para iluminar fotosensores individuais, gerando uma seqüência de sinais que identifica a posição exata do disco. A Figura 3.4 mostra um disco de um

encoder absoluto onde podemos perceber que cada posição tem uma combinação diferente.

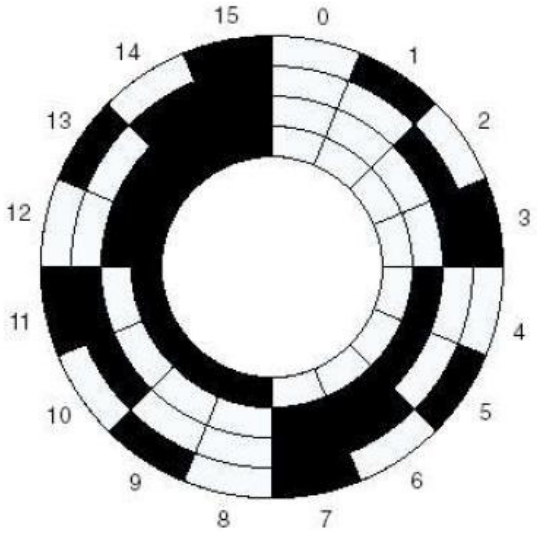


Figura 3.4 - Disco de um encoder absoluto
Fonte: Kilian (2000)

O número de trilhas determina quantas combinações diferentes será possível fazer, ou seja, qual é a resolução do sensor. Quanto mais trilhas tiverem melhor será a precisão embora isso também aumente o custo do dispositivo. A Figura 3.5 demonstra a influência do número de trilhas na resolução do encoder.

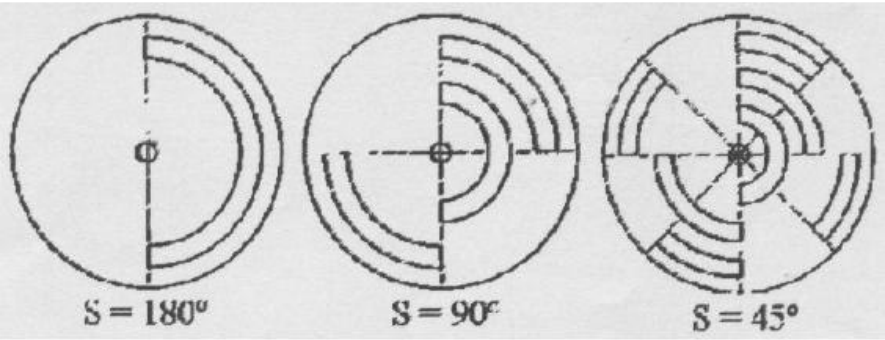


Figura 3.5 - Influência do número de trilhas na resolução do encoder
Fonte: http://www.ucs.br/ccet/demc/vjbrusam/inst/enc1.pdf

3.2 TACO-GERADORES

Taco-geradores ou tacômetros são transdutores de velocidade que atuam convertendo a velocidade de rotação de um eixo em tensão induzida. O taco-gerador consiste em um estator no qual está inserido um magneto permanente em um rotor no qual são enroladas espiras. O rotor disponibiliza tensão induzida, através das escovas, proporcionalmente à velocidade de giro do mesmo. (Albuquerque & Bastos, 1990)

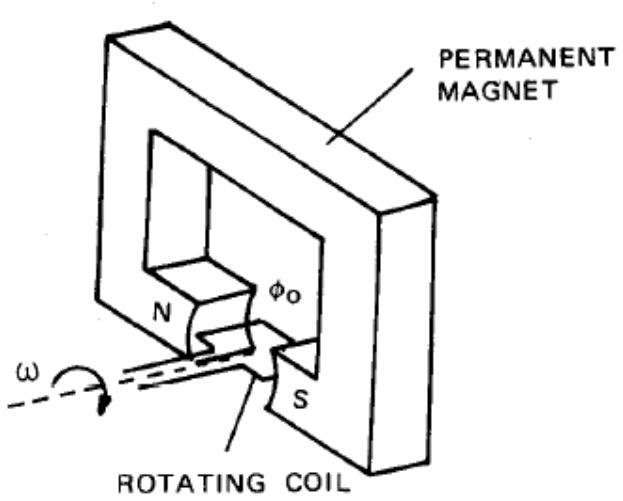


Figura 3.6 - Desenho esquemático de um taco-gerador.

Fonte: Albuquerque e Bastos (1990)

O campo magnético é obtido, em alguns casos, por meio de um imã permanente do tipo U, cujos pólos estão dispostos nas faces. Considere uma única espira girando à velocidade angular ω. O fluxo magnético da espira varia de acordo com a equação:

$$\emptyset = \emptyset_0 * \cos(\omega t) \qquad 3.1$$

Portanto, a tensão nos terminais da espira é:

$$e = -\frac{d\emptyset}{dt} \qquad 3.2$$

Substituindo 3.1 em 3.2, tem-se:

$$e = \emptyset_0 * \omega * \sin(\omega t) \quad 3.3$$

Onde o valor máximo é proporcional à velocidade angular. O parâmetro fundamental que caracteriza um taco-gerador é a constante taquimétrica (Kt), também chamada de gradiente taquimétrico. Esse coeficiente exprime a relação entre tensão de saída nos seus terminais e a velocidade de rotação.

$$Kt = E/\omega \quad 3.4$$

4 MICROCONTROLADORES

Os microcontroladores são dispositivos eletrônicos que consistem de muitos transistores, diodos e outros componentes interligados capazes de desempenhar muitas funções, apesar das dimensões extremamente reduzidas, que contêm um processador, pinos de entradas/saídas e memórias.

Através da programação dos microcontroladores pode-se controlar suas saídas, tendo como referência as entradas ou um programa interno. O que diferencia os diversos tipos de microcontroladores é a quantidade de memória interna (programa e dados), velocidade de processamento, quantidade de pinos de entrada/saída (I/O), alimentação, periféricos, arquitetura e conjunto de instruções.

Há pouco tempo, um microcontrolador se diferenciava de um microprocessador em vários aspectos. Primeiro e o mais importante, era a sua funcionalidade. Para que um microprocessador pudesse ser usado, outros componentes deveriam ser adicionados, tais como memória e componentes para receber e enviar dados. Por outro lado, o microcontrolador era projetado para ter tudo em apenas um dispositivo. Nenhum componente externo é necessário nas aplicações, uma vez que todos os periféricos necessários já estão contidos nele. Atualmente, as diferenças se resumem às velocidades de processamento, uma vez que já existem microprocessadores que possuem tais componentes e periféricos.

A velocidade de um microcontrolador é muito inferior à de um micro processador (podendo chegar a 1 milhão de vezes menor). Porém, há muitas aplicações que não necessitam de velocidades de processamento muito altas, e é neste contexto que se encontra a melhor situação para aplicação de um microcontrolador. O presente trabalho é um dos exemplos onde a utilização de um microcontrolador apresenta muitas vantagens em relação à utilização de microprocessador.

4.1 ESCOLHA DO MICROCONTROLADOR UTILIZADO NO TRABALHO

O projeto de controle de motores apresenta algumas necessidades que devem ser levadas em consideração no momento da escolha de um controlador ideal, que não seja inviável econômica e/ou tecnicamente, mas com funcionalidades suficientes que possibilitem a sua aplicação. As necessidades mais evidentes, e que resultaram na especificação do microcontrolador utilizado são:

- Possuir, no mínimo, um módulo de saída PWM;
- Possuir, no mínimo, cinco entradas com conversão A/D;
- Possuir, no mínimo, outros 21 pinos com saídas digitais;
- Ser um componente de fácil programação;
- Ser um componente viável economicamente.

O módulo de saída PWM e as entradas de conversão A/D se fazem necessários pelo escopo do projeto, uma vez que a atuação no motor DC foi feita utilizando a técnica de modulação por largura de pulso e a coleta dos dados do sinal de realimentação, a variação dos parâmetros de controle (Kp, Kd e Ki), e a variação do setpoint foram feitas utilizando a conversão analógica/digital.

Os 21 pinos com saídas digitais foram utilizados para operar os mostradores de 7 segmentos utilizados no projeto. A necessidade de se trabalhar com um componente de fácil programação se torna evidente devido ao fato dos autores do projeto não possuírem formação específica em linguagens de programação de microcontroladores ou linguagens de máquina. A viabilidade econômica está sempre presente em qualquer projeto de engenharia, uma vez que o custo sempre é um requisito de projeto.

Devido as necessidades de projeto acima apresentadas, foi escolhido o microcontrolador Microchip PIC 16F877A, que conta com a as características

necessárias à utilização. Seguem-se definições necessárias ao entendimento do projeto.

4.2 PIC

4.2.1 Definição

O PIC é um circuito integrado produzido pela Microchip Technology Inc. , que pertence a categoria dos microcontroladores, ou seja, um componente integrado que em um único dispositivo contem todos os circuitos necessarios para realizar um completo sistema digital programável.

O PIC pode ser visto externamente como um circuito integrado TTL ou CMOS normal, mas internamente dispõe de todos os dispositivos típicos de um sistema microprocessado, ou seja: Uma CPU (Central Processor Unit ou Unidade de Processamento Central) e sua finalidade é interpretar as instruções de programa; Memória PROM (Programmable Read Only Memory ou Memória Programavel Somente para Leitura) na qual irá memorizar de maneira permanente as instruções do programa; Memória RAM (Random Access Memory ou Memória de Acesso Aleatório) utilizada para memorizar as variáveis utilizadas pelo programa; Uma serie de pinos de I/O (entrada e saída) para controlar dispositivos externos ou receber pulsos de sensores, chaves, etc.; Uma serie de dispositivos auxiliares ao funcionamento, ou seja, gerador de clock, bus, contador, etc.

A presença de todos estes dispositivos em um espaço extremamente pequeno, dá ao projetista ampla gama de trabalho e enorme vantagem em usar um sistema microprocessado, onde em pouco tempo e com poucos componentes externos pode-se fazer o que seria oneroso fazer com circuitos tradicionais.

O PIC está disponível em uma ampla gama de modelos para melhor suprir às várias especificações dos mais variados projetos, diferenciando-se pelo

numero de pinos de I/O e pelo conteúdo do dispositivo. Inicia-se com modelo pequeno identificado pela sigla PIC12Cxx dotado de 8 pinos, até chegar a modelos maiores com sigla PIC17Cxx dotados de 40 pinos.

4.2.2 Memória

Segundo Souza (1971), Microcontroladores PIC são dotados de barramentos diferenciados para memórias de programa e dados. Além disso, há alguns modelos que possuem um terceiro tipo de memória, não volátil, chamada EEPROM. As memórias se encontram totalmente separadas na estrutura interna.

4.2.2.1 Memória de programa

Pode ser de 12, 14 ou 16, 18 ou 24 bits, e de vários tamanhos, dependendo do PIC a ser utilizado. Conhecida como PROM (memória somente de leitura programável), essa memória só permite, em geral, que dados sejam gravados uma vez. Entretanto, há modelos que contam com memórias EPROM (memória somente de leitura apagável) que podem ser apagadas, como no caso do PIC 16F877A utilizado no presente trabalho, ou apenas gravadas várias vezes com a exposição à luz ultravioleta e programada novamente por meio de gravadores específicos, por muitas vezes.

O primeiro endereço da memória de programa (0X00), que será executado ao rodar o PIC (inicialização ou reset), é para onde aponta o vetor reset. O vetor de interrupção, que será abordado adiante, é também armazenado na memória de programação e encontrado na posição 0x04.

4.2.2.2 Memória de dados

O PIC utiliza memória de acesso randômico, ou RAM, para armazenamento de dados, variáveis e registradores, utilizados no programa. É uma memória do tipo volátil, ou seja, as informações são perdidas ao se desligar o componente.

4.2.2.3 EEPROM

Modelos de atuais de PIC possuem uma memória não volátil, que mantém as informações mesmo sem alimentação. Essa memória é conhecida como EEPROM (memórias somente de leitura apagáveis eletricamente) e também pode ser utilizada para armazenamento de dados. Os modelos de PIC que não possuem esse tipo de memória podem se conectar a EEPROM externas utilizando pinos.

4.2.3 Interrupções

Há aplicações em que se necessita de que o controlador interrompa uma atividade que esteja sendo executada, devido a fatores internos ou externos, em determinado instante para efetuar outro tipo de atividade. Para tanto, um PIC dispõe de interrupções, com diferentes finalidades, que possibilitam essa mudança instantânea de atividade.

Quando uma interrupção ocorre, o microcontrolador pára a execução normal do programa e executa uma parte do programa designada como *Rotina de interrupção* (do inglês *Interrupt Service Routine*, ISR). Após executada a ISR, o programa volta à execução normal, de onde foi interrompido.

Segundo Ibrahim (2006), as interrupções possuem endereços fixos na memória do programa (endereços de vetor de interrupção) que podem ser únicos, ou, em alguns casos, há um vetor de endereço para cada fonte de interrupção.

4.2.3.1 Interrupção de timer

Normalmente utilizada para contagem de tempo. A interrupção ocorre quando um contador de tempo denominado TMR0 passa de 0xFF para 0x00. Esse contador TMR0 pode ser incrementado pelo clock do PIC ou por um sinal externo, dependendo da necessidade de aplicação no projeto.

4.2.3.2 Interrupção externa

Gerada por um sinal externo em porta (as) específica do PIC (dependendo do modelo pode apresentar mais de uma). Em muitos casos, a porta é a RB0. Para que um sinal externo seja processado imediatamente em um programa, esta porta deve ser previamente configurada como entrada no programa. Há a possibilidade de se configurar outras portas para que se interrompa o programa.

4.2.4 Portas

As portas I/O (portas de entrada/saída de dados) são o meio de comunicação do PIC com os dispositivos conectados ao mesmo. O PIC possibilita que todos os 33 pinos I/O contidos no modelo 16F877A (modelo utilizado no presente trabalho) sejam “configurados” como entradas ou saídas de dados, já que são bidirecionais. O modelo 16F877A conta com a porta A, com 6 pinos (RA0 a RA5), a porta B, com 8 pinos (RB0 a RB7), a porta C, com 8 pinos (RC0 a RC7), a porta D, com 8 pinos (RD0 a RD7), e a porta E, com 3 pinos (RE0 a RE2), segundo seu manual (Microchip PIC 16F87X Data sheet).

Cada porta bidirecional possui um registrador (TRIS) que configura os pinos como entrada ou saída. O valor enviado ao registrador é convertido em binário de modo a ser lido e, cada bit define um pino como entrada ou saída. Quando o

bit correspondente a um pino recebe o valor 0 o pino é definido como saída. Já, quando o bit recebe 1 o pino é definido como entrada.

Quando configurado como entrada, um pino pode receber dois tipos de dado. Se a tensão no pino for 5V, o pino entende como nível lógico alto ou 1. Já, quando a tensão no pino for 0V, o mesmo entende como nível lógico baixo ou 0. Na realidade, os níveis lógicos são definidos por uma faixa de tensão (chamado range), ou seja, para que um dado seja lido como nível lógico baixo, sua tensão não precisa, necessariamente de ser 0, porém algo em torno de 0. O mesmo acontece para nível lógico alto. Esse “range” de tensão depende do modelo do PIC.

Quando configurado como saída, o pino envia dois tipos de dados. Analogamente com a entrada, os dados enviados podem ser 0V, nível lógico baixo, ou 5V, nível lógico alto. Dessa forma, pode-se programar o microcontrolador para enviar dados de forma a fazer o mesmo a interagir com um dispositivo.

Há que se levar em conta esse valor da tensão aplicada em um pino configurado como entrada, pois, caso se aplique tensão superior a 5,5V, o pino será danificado, de acordo com seu manual (Microchip PIC 16F87X Data sheet).

O funcionamento descrito acima é aplicado quando se deseja interagir com circuitos digitais (que utilizam níveis lógicos alto e baixo), entretanto, não possibilita que um pino (ou uma porta) envie e receba sinais analógicos,embora estes sejam vastamente utilizados em sistemas, principalmente na medição de determinadas variáveis de um sistema.

No presente trabalho, necessita-se medir a velocidade do motor para que se possa controlá-lo e, como foi descrito, foi utilizado um taco-gerador que mede a velocidade do motor e envia ao controlador essa velocidade sob a forma de uma tensão. Esse é um exemplo onde não se pode aplicar uma leitura de sinal digital, pois o controlador só identificaria dois estados (0V e 5V) e há a necessidade de que ele identifique exatamente qual a tensão que nele está sendo aplicada.

4.2.4.1 Portas de conversão de dados Analógicos/Digitais

A solução desse problema se apresenta com a aplicação das portas de conversão A/D, que efetuam a conversão de um dado analógico (tensão de saída do taco-gerador) em um número inteiro de 10 bits (podendo ser de 0 a 1023), de acordo com o seu manual (Microchip PIC 16F87X Data sheet). No caso do PIC aplicado (16F877A), essa conversão se dá em apenas 8 pinos (os da porta A, exceto o RA4 e os da porta E).

O módulo de conversão A/D é descrito no manual (Microchip PIC 16F87X Data sheet). Verifica-se que este módulo possui 4 registradores, sendo 2 de configuração, ADCON0 e ADCON1, e 2 de resultados, ADRESH e ADRESL. O registrador de configuração ADCON0 controla a operação do módulo de conversão A/D (incluindo os canais de utilização e o clock de conversão). Já o registrador ADCON1 controla o funcionamento dos pinos (configurando como entradas analógicas, digitais ou tensão de referência, no caso de RA3).

Os registradores de resultado (ADRESH e ADRESL) recebem o valor lido pelo pino de conversão A/D. Esses registradores possuem 8 bits, cada e a concatenação dos mesmos gera o valor de 10 bits, excluindo os 6 mais significativos ou os 6 menos significativos, dependendo da configuração do registrador ADCON1.

Dessa maneira, pode-se coletar um valor de tensão entre 0 e 5V e transformar em um número que varia de 0 a 1023, ou seja, cada incremento no valor desse número corresponderá a um aumento de aproximadamente 4,9 mV na entrada do pino de conversão A/D, possibilitando a identificação da variação de uma tensão, não apenas a identificação de dois estados, como nas portas digitais.

Os valores coletados de tensão, dependendo da aplicação, podem precisar variar em um intervalo menor do que o valor usual, ou de inicialização (0V e 5V). Para isso, há a possibilidade de se configurar as tensões de referência (máximas e mínimas) para que esse intervalo de 1024 pontos (10 bits, ou 0 a 1023) esteja divido entre essas tensões.

4.2.4.2 Módulos de Captura, Comparação e PWM (CCP)

O microcontrolador PIC 16F877A possui duas saídas que podem ser configuradas para operar o módulo CCP (Captura, Comparação e PWM), são elas CCP1(RC2) e CCP2(RC3). Segundo Silva, R. (2007) Estes pinos realizam operações de captura de informações de 16 bits procedentes do registrador TMR1, comparação de um valor de registro com o TMR1 e modulação por largura de pulso com o registrador TMR2.

O registrador CCP1CON, de endereço na memória 17h, configura o pino CCP1 (pino RC2 com módulo CCP) para operar como modo de Captura, Comparação ou PWM. Este possui valores armazenados em dois outros registros de 8 bits, denominados CCPR1L (registro menos significativo) e CCPR1H (registro mais significativo), que compõe os 16 bits dos modos de comparação e captura. O registro de configuração CCP1CON está descrito pela Tabela 4.1.

Tabela 4.1 - Descrição do registrador CCP1COM

Endereço	Nome	Descrição
15h	CCPR1L	Byte LSB de captura, comparação e pwm
16h	CCPR1H	Byte MSB de captura, comparação e pwm

Registro de configuração CCP1CON endereço 17h.

Bit	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M	CCP1M0

Bit	Descrição	
CCP1M3, CCP1M2, CCP1M1, CCP1M0	0000	captura, comparação e pwm desligados
	0100	modo captura pulso descendente
	0101	modo captura pulso ascendente
	0110	modo captura a cada 4 pulsos ascendente
	0111	modo captura a cada 16 pulsos ascend.
	1000	modo comparação seta saída p/ CCP1IF
	1001	modo comparação limpa saída
	1010	modo comparação gera interrupção em CCP1IF, CCP1 não é afetado
	1011	modo comparação, gatilho especial seta CCP1IF e CCP1 reseta timer1 (TMR1)
	11xx	modo PWM

Fonte: Silva R. (2006)

4.2.4.3 Configuração do módulo PWM

A configuração do modo PWM utiliza como referência um temporizador, denominado timer2, que determina a freqüência do sinal PWM. Para devida explicação do modo, necessita-se, portanto, explicar o funcionamento deste temporizador e a sua atuação na freqüência do sinal.

O registrador TMR2 é um contador ascendente de 8 bits que produz uma saída EQ toda vez que a contagem de TMR2 coincide com o valor de PR2, utilizado como valor de referência (que apesar de poder ser alterado, é inicializado como FFh, ou 256). Os impulsos EQ são aplicados a um postscaler de 4 bits, que pode dividi-los em até 1:16, para se gerar uma interrupção (TMR2IF). A Figura 4.1 mostra a atuação do temporizador de modo simples para facilitar o entendimento do mesmo.

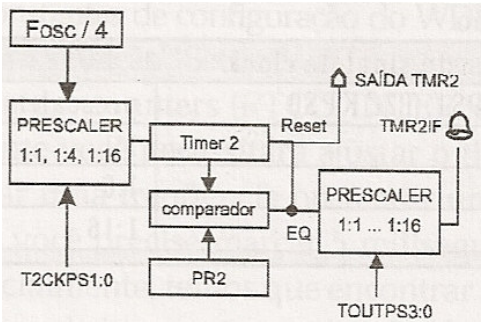


Figura 4.1 - Visualização do funcionamento do registrador Timer2

Fonte: Silva R. (2006)

A configuração desse temporizador é feita pelo registrador T2COM, descrito pela Tabela 4.2.

Tabela 4.2 - Descrição do registrador T2COM

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ToutPS3	ToutPS2	ToutPS1	ToutPS0	TMR2ON	T2CKPS	T2CKPS0

Bit	Descrição
TOUTPS3, TOUTPS2, TOUTPS1 e TOUTPS0	

TOUTPS3..TOUTPS0	postscaler
0000	1:1
0001	1:2
0010	1:3
0011	1:4
0100	1:5
0101	1:6
0110	1:7
0111	1:8
1000	1:9
1001	1:10
1010	1:11
1011	1:12
1100	1:13
1101	1:14
1110	1:15
1111	1:16

TMR2ON	Habilitação do timer 2 1 = habilitado 0 = desabilitado

Fonte: Silva R. (2006)

No compilador “CCS” o TMR2 é ajustado com a função “setup_timer_2(modo)”, onde o modo representa as constantes internas do compilador. As constantes servem para determinar o prescaler que servirá como um multiplicador para o clock escolhido, ou seja, se o prescaler estiver configurado como 1:2 o timer contará a cada 2 ciclos do clock. As configurações e constantes do timer 2 estão representadas na Tabela 4.3.

Tabela 4.3 - Descrição das constantes do timer 2 para o compilador “CCS”.

Constante	Descrição
RTCC_INTERNAL	Fonte de clock interno
RTCC_EXT_L_TO_H	Fonte de clock externo sensível à subida
RTCC_EXT_H_TO_L	Fonte de clock externo sensível à descida
RTCC_DIV_2	Prescaler 1: 2
RTCC_DIV_4	Prescaler 1:4
RTCC_DIV_8	Prescaler 1:8
RTCC_DIV_16	Prescaler 1:16
RTCC_DIV_32	Prescaler 1:32
RTCC_DIV_64	Prescaler 1:64
RTCC_DIV_128	Prescaler 1:128
RTCC_DIV_256	Prescaler 1:256

Fonte: Silva R. (2006)

Após o entendimento do funcionamento do registrador PR2 e do temporizador TMR2, segue-se agora a explicação do funcionamento do ciclo do PWM segundo Silva, A. (2007):

> Quando o valor do registro PR2 coincide com o do TMR2 a porta passa para 1, o valor de TMR2 é zerado, o registro CCP1 é setado e o TMR2 passa a ser comparado com o valor do CCPR1L. Quando o valor do TMR2 coincide com o do CCPR1L o latch da porta é resetado, gerando o ‘duty cycle’. O registro CCPR1H é usado para buffer interno apenas como leitura

A Figura 4.2 demonstra o funcionamento do TM2 sendo comparado com o PR2 e o CCPR1 de forma a facilitar o seu entendimento.

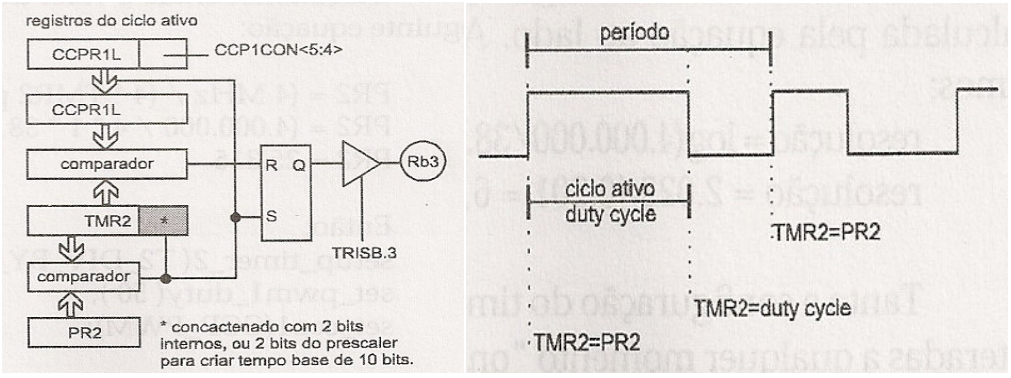


Figura 4.2 - Funcionamento dos registradores e determinação do ciclo ativo.

Fonte: Silva R. (2006)

A função que determina o ciclo ativo para o compilador “CCS” é chamada de “set_pwm1_duty(valor)”, na qual a variável valor corresponde a um número inteiro de 10 bits que representa o tempo do ciclo ativo. Já a função que inicializa o módulo CCP1 para operar no modo PWM no compilador “CCS” é chamada de “set_ccp1_(CCP_PWM)”, onde “CCP_PWM” é a constante que determina o modo como o pino CCP estará inicializado, ou seja, modo PWM.

4.3 LINGUAGENS DE PROGRAMAÇÃO

Como dito anteriormente, os microcontroladores PIC reúnem, em um único chip, todos os circuitos necessários para o desenvolvimento de um sistema digital programável. Entretanto, segundo Pereira (2003) a criação dos primeiros programas para microcontroladores era extremamente complexa, desgastante e morosa, além de necessitar de pessoas altamente capacitadas e, portanto de alto custo, pois tinham seus programas escritos em códigos chamados códigos de máquina, constituídos em dígitos binários.

Essa necessidade crescente de programação de sistemas levou ao desenvolvimento da linguagem Assembly, que consistem em apresentar os códigos de máquina usando mnemônicos, ou seja, abreviações de termos que descrevem a operação realizada pelo comando em código de máquina. Assim, após o desenvolvimento do programa em Assembly, é feita a conversão dos mnemônicos em códigos binários executáveis pela máquina. O programa que converte códigos em Assembly para códigos em linguagem de máquinas se chama Assembler.

Apesar da representação em Assembly ser muito mais simples do que a representação em linguagem de máquina, a utilização do Assembly não resolveu o problema dos programadores. Isso acontece, segundo Ferreira (2003), porque a linguagem Assembly é de baixo nível, ou seja, não possui comandos, instruções ou funções além dos definidos no conjunto de instruções do processador utilizado. Isso leva a trabalho extra ao programador, para

desenvolver rotinas e operação que não estão no conjunto de instruções do processador, e também produz programas muito extensos e complexos.

A solução desse problema passa pelas linguagens de alto nível, criadas para permitir a programação utilizando comandos de alto nível e que são posteriormente traduzidos para a linguagem de baixo nível (Assembly ou diretamente para código de máquina).

4.3.1 Linguagem “C”

A linguagem “C” foi desenvolvida nos laboratórios da Bell Labs, em 1972, A linguagem “C”, ou simplesmente C é uma linguagem de uso geral, ou seja, tem vasta aplicação na programação, pois possui recursos tanto de baixo quanto de alto nível e gera códigos eficientes e confiáveis (Pereira, 2003).

O “C” foi a linguagem escolhida para a programação do PIC 16F877A no desenvolvimento do trabalho devido aos seus recursos de alto nível, uma vez que a utilização de linguagens de baixo nível não pertence ao escopo do curso de Engenharia Mecânica e a adaptação dos autores do trabalho à este tipo de programação não seria viável. A programação em C também não pertence ao escopo do curso, entretanto, a adaptação dos alunos à sua utilização é consideravelmente mais simples pelo fato dos autores já estarem familiarizados com o uso de recursos disponibilizados pelas linguagens de alto nível.

5 CONTROLE

Os tipos de controle de mecanismos podem ser classificados de acordo com vários critérios independentes: em relação aos componentes utilizados na construção dos controladores, pode-se dividi-los entre analógicos e digitais, já em relação à realimentação, pode-se classificá-los como controladores de malha aberta ou de malha fechada. No caso de malha aberta, não há realimentação. Já no controle em malha fechada os sinais podem ser diversos e serão detalhados ao longo desse capítulo.

5.1 CONTROLE DE MALHA ABERTA

O controle de malha aberta se caracteriza por não apresentar realimentação. Nele não há nenhuma informação a respeito do que está, de fato, acontecendo com o mecanismo controlado, ou seja, o sinal de controle aplicado ao processo em um determinado instante não depende da evolução desse processo ou de como ele está se comportando, pois a saída não é comparada com a entrada de referência (BALL, 2004; OGATA, 2005). É considerado por Ball (2004) como o mais simples de todos os tipos de controle, entretanto Ogata (2005) destaca que “[...] a precisão do sistema depende de uma calibração” e que “na presença de distúrbios, um sistema de controle de malha aberta não vai executar a tarefa desejada”.

Um exemplo de controle em malha aberta citado por Ball (2004) é o controle da velocidade de um motor vibratório de um celular ou pager. Nesse caso, não faz diferença para o usuário ou para o próprio instrumento se a velocidade do motor variar 10% ou 20% e, portanto o microprocessador pode apenas mandar o sinal para o motor sem monitorar a velocidade real que está sendo desenvolvida, que dependerá de diversos fatores como o estado das escovas do motor e a condição da bateria. Podemos também usar como exemplo o sistema da máquina de lavar roupas, onde as funções colocar de molho, lavar

e enxaguar são executadas apenas em função do tempo e de uma seqüência previamente definida e não há medição do sinal de saída.

Nesses exemplos, fica clara a característica essencial do controle de malha aberta: sua entrada não depende de sua saída, entretanto, cabe destacar que esse tipo de controle apresenta melhores resultados quando é usado em sistemas previsíveis, sem presença de distúrbios.

5.2 CONTROLE DE MALHA FECHADA

Também chamados de controle com realimentação, esses sistemas utilizam o sinal de saída para determinar o sinal de entrada. Ogata (2005) define que “em um sistema de controle de malha fechada, o sinal de erro atuante, que é a diferença entre o sinal de entrada e o sinal de realimentação [...], realimenta o controlador, de modo que minimize o erro e acerte a saída do sistema ao valor desejado.”.

Esse tipo de controle pode ser exemplificado através de um sistema de aquecimento ou resfriamento de ambiente. Esses mecanismos medem a temperatura real, comparam com a temperatura de referência estabelecida pelo usuário e usam essa diferença para desligar, ligar ou definir a potência do equipamento de aquecimento/resfriamento, de forma que independentemente das condições exteriores, a temperatura interna permanece próxima daquela de referência (BALL, 2004; OGATA, 2005).

Ogata (2005) cita ainda o corpo humano como um exemplo muito interessante de controle com realimentação, evidenciando o fato que sistemas de controle não são exclusividade da engenharia, podendo ser encontrados em várias outras áreas. Segundo o autor, a temperatura corporal e a pressão sanguínea são mantidas constantes por meio da realimentação de ordem fisiológica e é isso que faz com que o corpo humano seja relativamente insensível a perturbações externas. Essa realimentação exerce uma função vital, pois permite o perfeito funcionamento do corpo humano nos casos de mudança de

ambiente. Na engenharia, a realimentação também é vital para o controle de mecanismos que estarão sujeitos à mudanças de ambiente, chamadas de distúrbios.

Basicamente, o controle com realimentação não seria necessário se o mundo fosse completamente previsível, pois sua principal característica é fazer com que o sistema responda a eventos incertos. Por outro lado, essa que, a princípio, se destaca como grande qualidade do controlador de malha fechada não possibilita que *ações de antecipação* sejam tomadas visto que esse tipo de sistema atua reativamente, o que quer dizer que o controle só será possível quando o efeito da perturbação se manifestar na variável controlada (CARVALHO, 2000).

O controle de malha fechada pode ser feito usando apenas um amplificador operacional, entretanto ele se torna muito mais interessante e abrangente com uso de microprocessadores, que permitem gerar um sinal de saída mais complexo e útil, baseados não apenas na entrada e saída, mas também no histórico de como o sistema reage às mudanças, na carga aplicada e na taxa de variação da resposta. O trabalho apresentado utiliza o PIC® 16F877A como microcontrolador e implementa um controlador do tipo PID, que será explicado mais adiante.

5.2.1 Controle ON/OFF

O princípio de funcionamento do controle ON/OFF é extremamente simples. O sistema avalia se a saída está acima ou abaixo da referência, se estiver acima, o controlador desliga o equipamento e no caso contrário, quando a saída está abaixo da referência, liga-o novamente. É comum, embora não seja obrigatório, o sinal de saída exceder o sinal de referência consideravelmente no início do processo, comportamento chamado de overshoot, devido à inércia do sistema. Além disso, com esse tipo de controle, a saída do sistema sempre irá oscilar em torno do ponto de referência (BALL, 2004) e o valor médio da variável controlada terá um desvio residual, chamado de erro de offset. A Figura 5.1

mostra o comportamento da variável controlada e da variável manipulada em um controle do tipo ON/OFF.

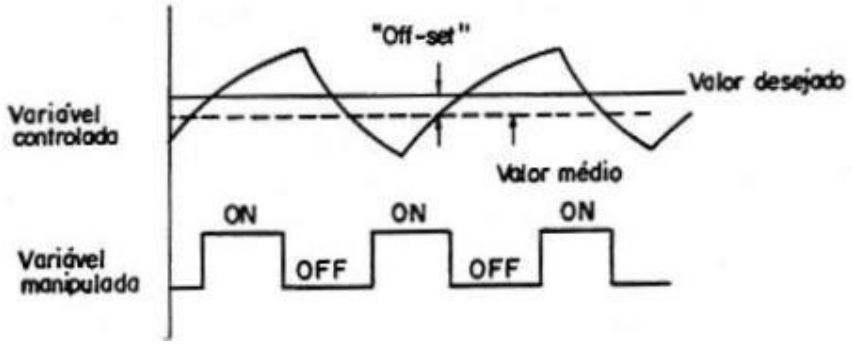


Figura 5.1 - Variável controlada e variável manipulada em um controle ON/OFF
Fonte: Oliveira (1999)

De acordo com Ball (2004), os controladores ON/OFF funcionam melhor quando o objeto controlado não responde rapidamente a mudanças no sinal de controle e quando o sensor que mede o estado do objeto controlado responde a mudanças muito mais rápido do que o objeto controlado em si. Apesar de suas restrições, controladores ON/OFF são amplamente usados em nosso dia-a-dia por serem econômicos. Os controles de nível de água através de “bóias” e termostatos de geladeira são ótimos exemplos.

5.2.2 Controle Proporcional

O conceito básico do controle proporcional é o de variar o sinal de controle baseando-se na amplitude da diferença entre a condição desejada (referência) e a real. Esse valor, chamado de erro, é o que define o sinal de saída, por meio de um ganho (BALL, 2004). A saída de um controlador proporcional é dada pela fórmula:

$$u(t) = Kp \times e(t) \qquad 5.1$$

Onde *e* é o erro e Kp é o ganho proporcional.

Assim como no controle ON/OF, aqui o sinal de controle a ser aplicado será positivo caso o valor de saída do processo seja menor do que a referência e negativo caso o valor de saída do processo seja maior do que a referência. A diferença é que o sinal será proporcional ao módulo do erro, ou seja, não temos apenas as condições ligado ou desligado, como no ON/OFF, mas a potência do atuador também irá variar de acordo com a "distância" entre o ponto que se deseja e a situação real.

Ball (2004) destaca duas vantagens do controle proporcional em relação ao ON/OFF:

- Menor oscilação e overshoot;
- Possibilidade de ajustar o sinal de controle de acordo com as características do objeto a ser controlado, através da variação do coeficiente de proporcionalidade, o ganho.

Embora o controle proporcional elimine o problema da oscilação, ele não consegue eliminar completamente o erro de offset. A Figura 5.2 mostra o resultado de um controle pela ação proporcional em relação a um distúrbio, nela pode-se verificar a diminuição da oscilação se comparado ao controle ON/OFF e a existência do erro residual. De acordo com Carvalho (2000), quanto maior o ganho, menor será esse erro no regime permanente, o que representa uma precisão melhor do controle, entretanto, o comportamento no regime transitório se torna mais oscilatório com o aumento do ganho, ou seja, leva a um tempo maior de acomodação, além disso, na maioria dos processos físicos, aumentar excessivamente o ganho proporcional pode levar o sistema à instabilidade.

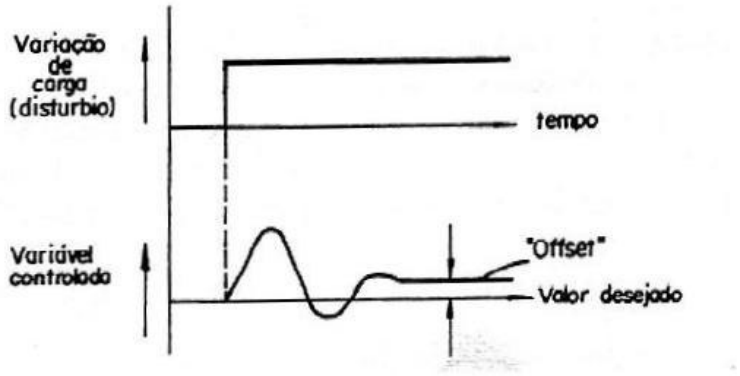


Figura 5.2 - Resposta do controle pela ação proporcional
Fonte: Oliveira (1999)

Em alguns casos, uma função de transferência que inclua um ganho é necessária se o atuador precisar se manter funcionando em uma determinada potência mesmo quando o erro for próximo ou igual a zero. Um exemplo é um motor que deve girar a uma velocidade pré-determinada, caso a velocidade real seja igual à desejada, ainda assim a alimentação do motor deve fornecer energia suficiente para manutenção dessa condição. Nesse caso, a equação que rege o controle é:

$$u(t) = \big(Kp \times e(t)\big) + M \qquad 5.2$$

Onde M é um ganho.

O problema do controlador proporcional é que ele ajusta o sinal de saída baseado somente na diferença entre o valor medido e o valor desejado. Não há possibilidade de considerar condições “desconhecidas” para o microprocessador. Para o controle da velocidade de cruzeiro de um automóvel, por exemplo, é interessante considerar se o mesmo está em uma subida, descida ou se há perda de potência devido a algum outro fator, como ligar o ar-condicionado. O controle proporcional puro não é adequado para essa situação, pois a quantidade da mistura ar-combustível que deve ser fornecida ao motor em uma subida íngreme não é a mesma que em uma descida, mesmo que a diferença entre a velocidade real e a de referência seja a mesma nos dois casos (BALL, 2004).

5.2.3 Controle Derivativo

A derivada é, essencialmente, uma medida de quão rápido uma função está variando. Matematicamente falando, é a inclinação da curva. Baseado nesse conceito, o objetivo do controle derivativo é incorporar uma maneira de controlar melhor cargas variáveis ou desconhecidas uma vez que ele é capaz de avaliar o quanto uma medida está variando.

A ação derivativa é antecipatória ou preditiva e tende a fazer com que o sistema reaja mais rapidamente, pois é uma tendência baseada na evolução do erro. Além disso, a implementação analógica de um *derivador puro* é fisicamente impossível. De fato, derivativo não é, isoladamente, uma técnica de controle, ele "é sempre combinado, pelo menos, com a ação proporcional" (CARVALHO, 2000). Um termo derivativo é adicionado à equação do controle proporcional já mencionada anteriormente, de forma que se o erro estiver diminuindo, a inclinação negativa da curva fará com que a derivada seja negativa e o sinal de saída seja menor e se o erro estiver aumentando, a derivada será positiva e a saída será maior (BALL, 2004):

$$u(t) = Kp \times e(t) + Kd\frac{de(t)}{dt} \qquad 5.3$$

Esse efeito é mostrado na Figura 5.3.

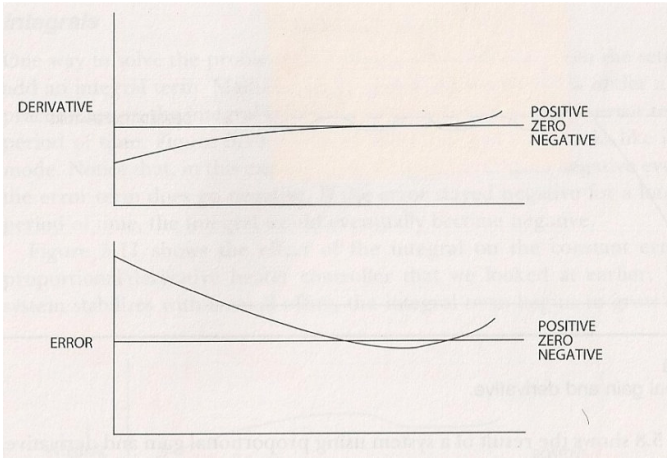


Figura 5.3 - Termo derivativo e sua influencia no erro

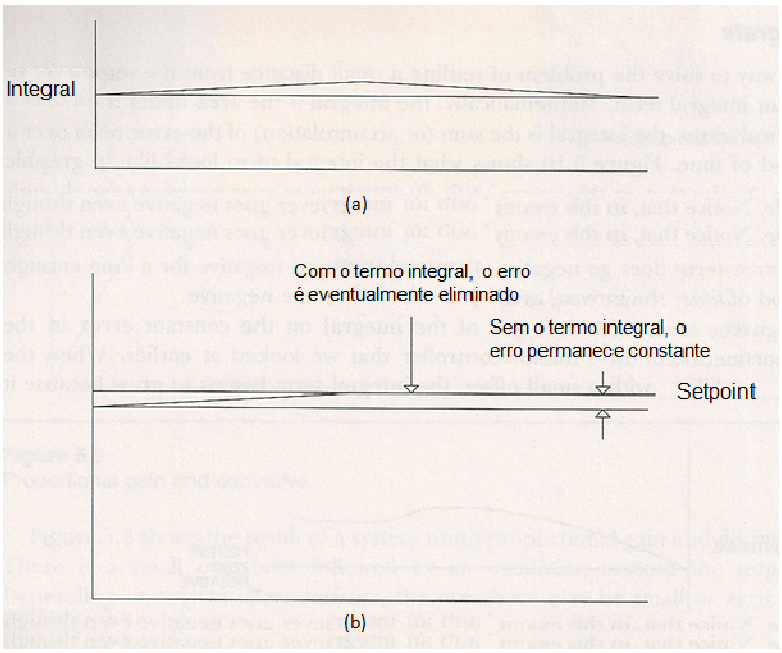
Fonte: Ball (2004)

Cabe destacar um ponto citado por Carvalho (2000), Ball (2004) e Ogata (2005), que é o fato da derivada ser zero para um valor de erro constante, mesmo que esse erro seja grande. Isso implica dizer que o controle derivativo só atua quando há variação no erro, ou seja, durante o regime transitório. Além disso, Ball (2004) enfatiza que esse controle não elimina o erro permanente, visto que quando a medida real se aproxima da referência, a inclinação da curva do erro se torna cada vez menor e o termo derivativo tende a zero, deixando de exercer influência sobre o controle.

Outro problema é que saltos bruscos são incluídos no processo toda vez que existe uma alteração em degrau do setpoint. Para evitar esse problema, a maioria dos controladores atuais apresentam a opção de usar a derivada da variável controlada (BALL, 2004). Segundo Ball (2004) “basicamente atinge-se a mesma resposta às perturbações [...], mas as variações mais bruscas na referência são suavizadas (filtradas) pelo processo antes de chegar ao modo derivativo”.

5.2.4 Controle Integral

Matematicamente, a integral é a área abaixo de uma curva. Na prática, a integral é o valor acumulado de uma função (no caso do controle, o erro) durante um período de tempo e por isso a ação de controle integral é usada para eliminar o offset, citado anteriormente como um dos maiores problemas envolvendo o controle proporcional e derivativo. Quando o sistema estabiliza com um erro constante, o termo integral começa crescer e após um determinado tempo, a integral atinge um valor grande o suficiente para afetar a saída do sistema, fazendo com que esse sinal se aproxime da referência, figura 5.4 (BALL, 2004).



5.4 - (a) Integral do erro e (b) eliminação do offset pelo termo integral
Fonte: Ball (2004)

Um problema que pode surgir com a ação de controle integral é quando o microprocessador exige uma saída maior do que o sistema consegue gerar, como no caso de um aquecedor trabalhando em um ambiente muito maior do que o projetado, e por isso o sistema não consegue responder tão rápido quanto faria em uma situação normal, com isso o erro acumulado se torna muito grande. Assim, quando o valor de referência for atingido e o erro se aproximar de zero, os controles proporcional e derivativo deixarão de atuar e o controle integral fará com que a saída continue sendo “corrigida”, afastando-a da referência (BALL, 2004; CARVALHO, 2000). Esse efeito é conhecido como *Wind-up* e deve ser evitado. Uma maneira de eliminá-lo é inserir um limite máximo para o termo integral.

5.2.5 Controle PID

O controle PID é a junção das ações de controle proporcional, integral e derivativo. De acordo com Ogata (2005), essa ação combinada tem as

vantagens individuais de cada uma das três ações de controle. O termo proporcional faz o sinal de saída seguir o sinal de entrada (referência), o termo derivativo permite que o sinal de saída responda a rápidas mudanças na entrada e se adéqüe a diferentes tipos de cargas e por último, o termo integral compensa erros acumulados.

A equação básica do controlador PID é:

$$u(t) = Kp\left(e(t) + \frac{1}{Ti}\int e(t)dt + Td\frac{de(t)}{dt}\right) \quad 5.4$$

Ou ainda:

$$u(t) = Kp \times e(t) + Ki\int e(t)dt + Kd\frac{de(t)}{dt} \quad 5.5$$

5.2.5.1 Sintonia de controladores PID

É chamado de sintonia do controlador o processo de selecionar parâmetros que irão garantir uma dada especificação de desempenho.

Para uma planta que o modelo matemático pode ser obtido, existem várias técnicas de projeto para determinar os parâmetros do controlador que vão impor as especificações do regime transitório e do regime permanente do sistema de malha fechada. Entretanto, muitas vezes a planta é muito complexa, de modo que seu modelo matemático não pode ser obtido facilmente, por isso a abordagem analítica do projeto do controlador PID não é possível e se faz necessário recorrer a abordagens experimentais de sintonia (OGATA, 2005).

Ziegler e Nichols propuseram dois métodos através dos quais pode-se determinar os valores do ganho proporcional Kp, do tempo integral Ti e do tempo derivativo Td baseando-se na resposta temporal da planta (OGATA, 2005).

O primeiro método de Ziegler-Nichols consiste em obter a resposta da planta a um degrau, que pode ser gerada experimentalmente ou através de uma simulação dinâmica da planta. A curva em S, normalmente gerada, pode ser

caracterizada por duas constantes: o atraso e a constante de tempo. A função de transferência pode ser aproximada por um sistema de primeira ordem com um atraso de transporte, pela equação 5.6.

$$G(s) = \frac{k}{\tau s+1} e^{-td*s} \qquad 5.6$$

Onde:

k é o ganho estacionário

τ é a constante de tempo

e t_d é o atraso.

Ziegler e Nichols sugeriram escolher os valores de Kp, Ti e Td de acordo com a fórmula que aparece na tabela 5.1.

Tabela 5.1 - Regra de sintonia de Ziegler-Nichols baseada na resposta ao degrau da planta (primeiro método)

Tipo de controlador	Kp	Ti	Td
P	$\frac{\tau}{t_d}$	∞	0
PI	$0{,}9\frac{\tau}{t_d}$	$\frac{t_d}{0{,}3}$	0
PID	$1{,}2\frac{\tau}{t_d}$	$2\ t_d$	$0{,}5\ t_d$

No segundo método, define-se primeiro Ti = ∞ e Td = 0 e aumenta-se Kp de 0 até o valor crítico K_{cr}, no qual a saída exibe uma oscilação sustentada pela primeira vez, determinando-se, dessa maneira tanto o K_{cr} quanto o período correspondente P_{cr}. Os valores de Kp, Ti e Td sugeridos por Ziegler-Nichols estão na tabela 5.2.

Tabela 5.2 - Regra de sintonia de Ziegler-Nichols baseada no ganho crítico K_{cr} e no período crítico P_{cr} (segundo método)

Tipo de controlador	Kp	Ti	Td
P	$0,5K_{cr}$	∞	0
PI	$0,45K_{cr}$	$\frac{1}{1,2}P_{cr}$	0
PID	$0,6K_{cr}$	$0,5P_{cr}$	$0,125P_{cr}$

5.3 CONTROLE ANALÓGICO X DIGITAL

Como citado anteriormente, os sistemas de controle podem ser analógicos ou digitais, dependendo dos componentes com os quais foram construídos.

Os controladores analógicos, ou contínuos, são aqueles que, como o próprio nome já sugere, são construídos a partir de partes analógicas como resistores, capacitores e amplificadores operacionais (FRANKLIN, POWELL & WORKMAN, 1998).

Os controladores digitais são os que usam microcontroladores para gerar o sinal de saída e sua grande vantagem é que a lógica digital é muito mais fácil de entender e muito mais flexível do que seu equivalente analógico, o que permite alterações e adaptação de maneira muito mais rápida. Por outro lado, esses controladores trabalham com amostras, que não são adquiridas continuamente. Com isso, qualquer informação que varie mais rápido do que a taxa de amostragem será perdida. Além disso, não é possível integrar utilizando sistemas digitais, portanto para resolver equações diferenciais é necessário que aproximações sejam feitas para transformar essas equações em equações algébricas, com apenas somas e produtos, chamadas de equações de diferenças.

Para digitalizar um sinal analógico e assim poder enviá-lo ao microcontrolador é necessário um conversor A/D, que possui um dispositivo de amostragem

(sampler) no qual a cada T segundos uma chave se fecha e permite a aquisição do sinal. Esse tempo T é chamado de tempo de amostragem e 1/T é a frequência de amostragem em ciclos por segundo (Hz). Além disso, também é preciso converter o sinal físico, normalmente uma tensão, em um número binário, geralmente de 10 ou 12 bits (FRANKLIN, POWELL & WORKMAN, 1998). No caso desse projeto, a porta A/D utiliza uma variável de 10 bits para receber esse valor, que então poderá ser de 0 a 1023, resultando em uma resolução de 0,1%. Os diagramas de blocos para um sistema analógico e digital são mostrados na Figura 5.5 onde podemos destacar o sampler e o conversores A/D e D/A.

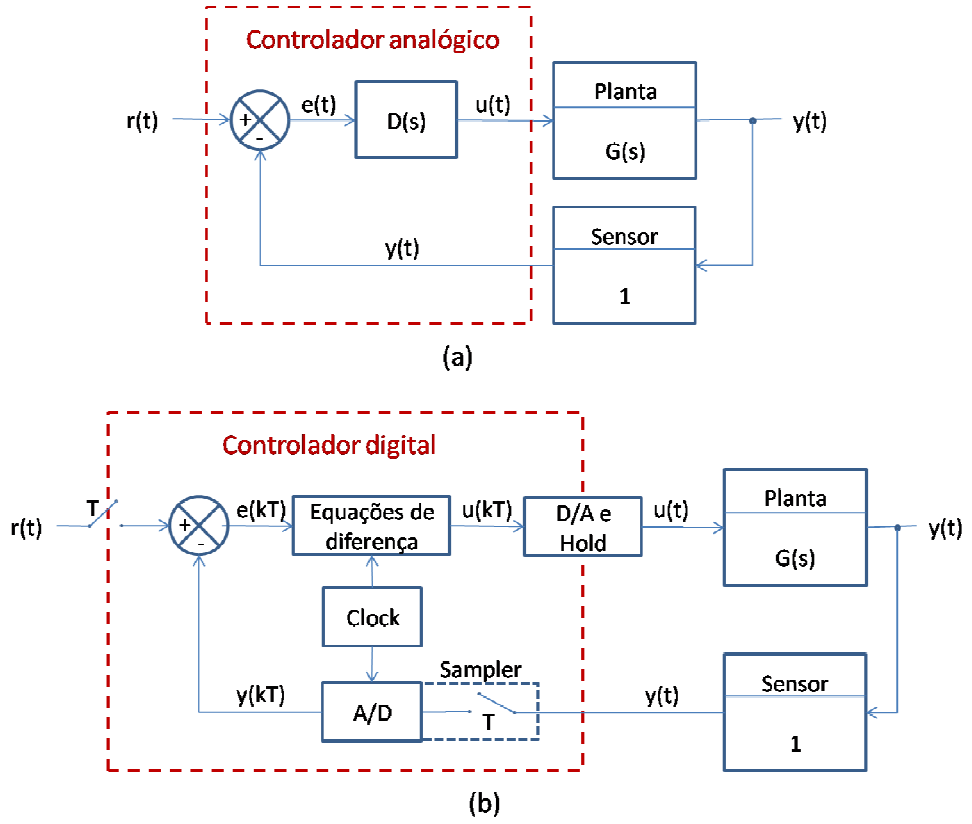
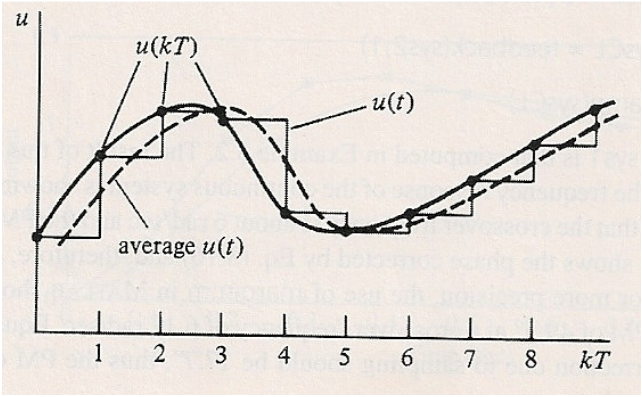


Figura 5.5 - Diagrama de blocos de (a) um controlador analógico e (b) um controlador digital
Fonte: Adaptado Franklin, Powell & Workman

O efeito da amostragem, causado pelo atraso associado ao sample and hold, é considerado por Franklin, Powell & Workman (1998) como o mais notável impacto da utilização de sistemas digitais. Um atraso em qualquer sistema

realimentado diminui a estabilidade e o amortecimento do mesmo devido ao fato do valor lido em um determinado momento ser mantido constante até que o novo valor seja disponibilizado pelo processador. Na figura 5.6 pode-se verificar esse efeito. Nela a linha contínua representa o sinal real, *u(kT)* são os valores obtidos em cada leitura e "travados", portanto *u(T)* é formado por degraus e a sua média, representada pela linha pontilhada tem um atraso de T/2 em relação à *u(kT)* .



5.6 - Efeito da amostragem
Fonte: Franklin, Powell & Workman (1998)

6 PWM

Há inúmeras aplicações na engenharia em que se faz necessário controlar a potência de uma carga qualquer (lâmpadas, aquecedores, motores, etc.). No desenvolvimento do presente trabalho propõe-se a utilização da técnica de controle de motores DC por largura de pulso, ou *pulse widith modulation* (PWM).

Para que se faça o controle da velocidade de, por exemplo, um robô móvel, necessita-se então de controlar a velocidade de giro dos motores DC que estão acoplados às suas rodas. De acordo com Braga, N. (2009), basicamente são dois os tipos de controles para as cargas mencionadas acima: O controle linear (mais simples e tradicional) e o controle pulsante (onde se classifica o controle PWM usado no projeto).

6.1 CONTROLE LINEAR DE POTÊNCIA

A maneira mais simples de se controlar uma carga de potência é utilizando um reostato em série com a carga, conforme Figura 6.1 (Ghirardello, 200-?). Ao se variar a resistência do reostato, altera-se a corrente que passa pela carga e, assim, a potência consumida pela mesma.

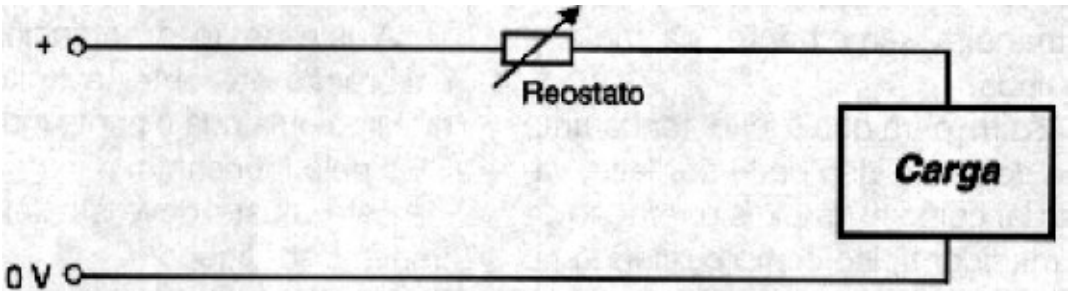


Figura 6.1 - Controle de carga utilizando potenciômetro

Fonte: Ghirardello (200-?)

A desvantagem desse tipo de controle é a potência dissipada, dada pela multiplicação da corrente pela queda de tensão no reostato, que é consideravelmente alta (podendo chegar a ser maior que a potência fornecida à carga), prejudicando a eficiência do sistema.

Há circuitos que se valem da utilização de transistores de potência para reduzir a corrente no reostato, e a potência por ele dissipada. Mas, apesar dessa redução, a potência dissipada pelo dispositivo que controla a corrente é elevada.

A preocupação com a redução de perdas nos circuitos e a extinção de grandes dissipadores de calor (reduzindo o espaço) nos leva a utilizar métodos mais adequados e eficazes para efetuar o controle da potência fornecida à carga, como o PWM, utilizado no presente trabalho.

6.2 CONTROLE PULSANTE DE POTÊNCIA

Conforme Guerreiro,J. ET AL (2003) a modulação por largura de pulso, do inglês *Pulse Width Modulation, (PWM)* é uma técnica poderosa de controle de circuitos analógicos a partir de circuitos digitais

Sinais analógicos são sinais que variam no tempo com resolução infinta. Sinais digitais são sinais que não variam com resolução infinita, mas são “discretizados”, assumindo certos valores (geralmente 0 a 5V).

O PWM é uma maneira de codificar sinais analógicos com sinais digitais. Ghirardello, A explica o funcionamento da técnica do PWM partindo de um

circuito formado por um interruptor de ação rápida conforme a Figura 6.2

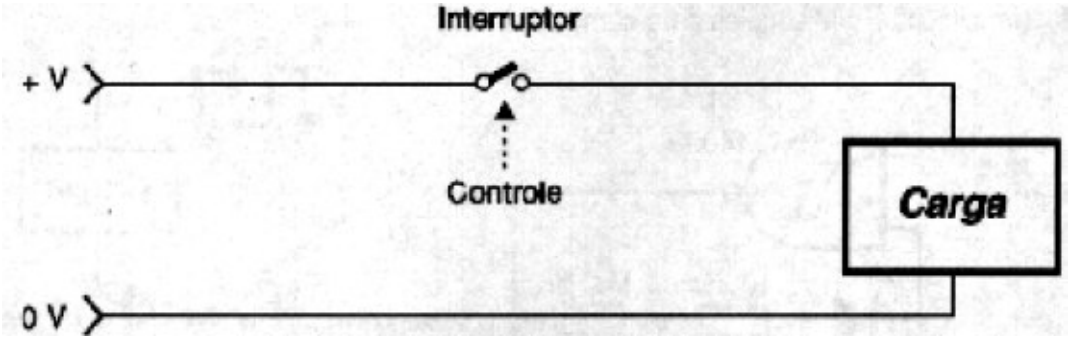


Figura 6.2 - Controle pulsante de carga

Fonte: Ghirardello (200-?)

Na posição aberta, o interruptor não libera o fornecimento de potência à carga. Ao ser fechado, a carga recebe potência máxima da fonte. Ao abrir e fechar o interruptor de modo a manter 50% do tempo fechado e 50% aberto, em média, a potência média fornecida será a metade da potência máxima. A Figura 6.3representa o funcionamento do PWM.

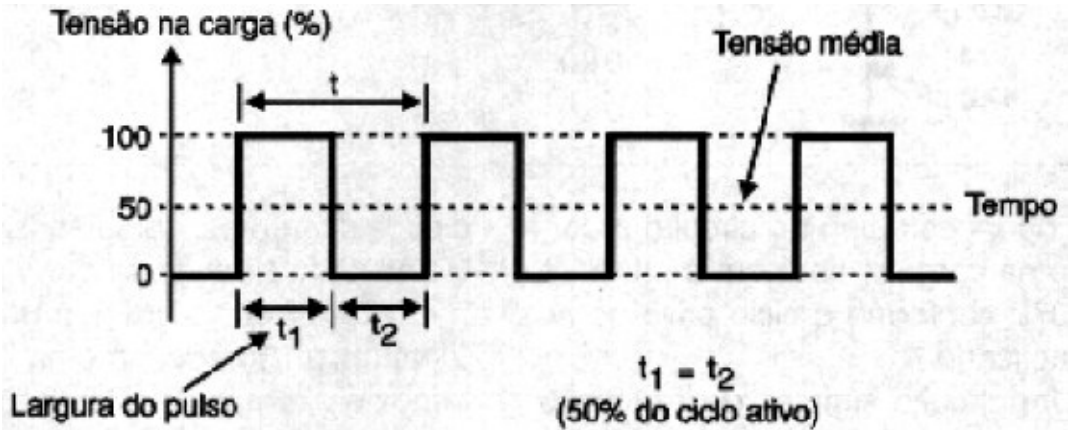


Figura 6.3 - Sinal PWM transmitido à carga (50% de duty cycle).

A relação entre o tempo em que o interruptor permanece aberto e o tempo de um ciclo (tempo aberto + tempo fechado) é chamada de *duty cycle*. Variando-

se o *duty cycle* (ou a largura do pulso), a potência média fornecida à carga varia proporcionalmente, como mostra a Figura 6.4

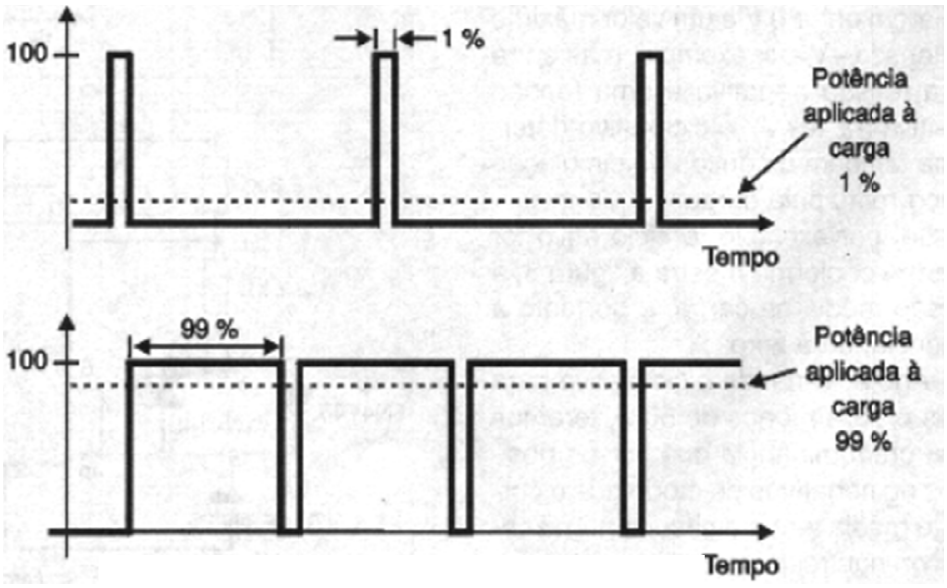


Figura 6.4 - Efeito da alteração do duty cycle na potência do motor.

Fonte: Ghirardello (200-?)

Os sinais acima são sinais PWM com 1% e 99% de *duty cycle*, ou seja, no primeiro a carga trabalha com 1% da sua potência máxima e no segundo com 99%.

Segundo Braga, N. (2009) o PWM é excelente para controlar a velocidade do motor. Isso por que mesmo aplicando a tensão máxima do motor, como sua duração é pequena, ocorre uma diminuição da tensão e corrente médias, o que reduz a potência fornecida sem alterar a tensão aplicada nos motores.

A importância de não se alterar a tensão de alimentação para reduzir a potência fornecida se dá quando se necessita de baixas velocidades de rotação. Quanto mais baixa a tensão de alimentação, menor o torque do motor e mais difícil será vencer a inércia inicial do sistema. Assim, o controle linear gera muitas dificuldades para trabalhar a baixas tensões, limitando a velocidade mínima de trabalho.

Mesmo a velocidades muito baixas, o controle PWM envia a tensão máxima ao motor e, com isso, o torque não se altera. Além disso, a corrente durante o intervalo de tempo do pulso também é máxima e isso garante que seja suficiente para tirá-lo da imobilidade e manter o seu torque.

Há um parâmetro importante na utilização da técnica de controle por largura de pulso, a freqüência do PWM. Essa freqüência é que determina quanto tempo vai durar o ciclo do PWM. O duty cycle determina apenas o percentual do tempo que a saída PWM estará no ciclo ativo.

Esse tempo de duração do ciclo total do PWM influencia alguns parâmetros como a vida útil do motor e a velocidade de chaveamento do transistor (e por tanto o calor gerado nesse componente). Quando muito baixa, a freqüência pode gerar “solavancos” nos momentos de entrada e saída do ciclo ativo, aumentando a vibração do motor. Além disso, a freqüência pode fazer com que o motor vibre dentro da faixa de percepção sonora humana, podendo causar incômodos devido a barulhos.

O período do PWM utilizado no trabalho foi de 4,1ms e se percebeu que este atende às necessidades do projeto. Portanto, como este não era o objetivo do trabalho (além de ser motivo para muito estudo especializado), o período não foi otimizado.

7 METODOLOGIA

7.1 DESCRIÇÃO DO ENSAIO

Desde o início do trabalho se verificou a necessidade de aprimorar os conhecimentos, teóricos e práticos, dos autores referentes a áreas onde a formação em engenharia mecânica não seria suficiente para o desenvolvimento do mesmo, uma vez que alia conceitos em áreas do conhecimento de engenharia elétrica e programação. Portanto, foi necessário empregar grande parte do tempo do desenvolvimento para efetuar tarefas simples, mas que serviram de suporte e possibilitaram a utilização de ferramentas e dispositivos complexos.

Para alcançar o nível de complexidade atingido no trabalho, foi necessária a familiarização com circuitos elétricos mais avançados que empregavam transistores, diodos, amplificadores operacionais e microcontroladores,

Uma das grandes dificuldades que surgiu durante a elaboração do estudo foi a coleta dos sinais elétricos e seu devido tratamento, necessários para caracterizar os motores utilizados e analisar o comportamento da velocidade em função do sinal de atuação.

Como descrito no capítulo de medição de velocidades, uma das possibilidades de medição seria utilizando encoders, que já gerariam sinais digitais provenientes de fototransistores. Foram efetuados testes utilizando encoders provenientes de mouses antigos, uma vez que a aquisição do mesmo no mercado seria inviável economicamente.

A leitura dos sinais provenientes dos encoders não foi satisfatória. Os encoders utilizados em mouses não podiam ser utilizados, pois o emissor dos sinais luminosos, captados pelo fototransistor, emitia, na realidade, sinais em forma de pulsos pois o mouse utiliza a diferença entre a captação dos pulsos para determinar qual o sentido de giro que se está empregando ao encoder.

Além disso, houve a constatação de que os encoders retirados dos mouses seriam muito lentos, uma vez que trabalham a velocidades muito mais baixas que as velocidades atingidas por um eixo de motores elétricos, mesmo que de baixa potência, uma vez que possuem velocidades superiores a 2000 rpm.

Outra possibilidade de medição da velocidade do motor seria a utilização de um taco-gerador que, conforme abordado no capítulo referente à medição de velocidades, converte a velocidade do eixo em um nível de tensão que pode ser lido para que se determine a sua velocidade através da curva de Tensão por Velocidade.

A escolha do taco-gerador como medidor de velocidade foi ainda motivada pela utilização de um kit disponibilizado pelo LECO (Laboratório de Ensino de Controle do Departamento de Engenharia Elétrica da UFES), que conta com um motor acoplado a um taco-gerador e um sistema de aquisição de dados em tempo real, para efetuar estudos e ensaios relevantes ao controle do motor.

Tendo à disposição o kit do LECO, foi desenvolvido um circuito capaz de efetuar a leitura do taco-gerador e controlar o motor com base nos dados provenientes dessa leitura. Primeiramente o circuito foi montado em protoboard a fim de que se realizassem testes para validação do programa e dos dispositivos.

O circuito projetado é composto por um transistor (TIP 122) capaz de fornecer, quando excitado por um sinal de tensão na base adequado, satura, chaveando a alimentação necessária para movimentar o motor DC. O sinal que excita a base é proveniente do módulo PWM do PIC.

A limitação do sinal de entrada no PIC gera alguns problemas para a configuração do circuito. Além da baixa tensão e correntes de saída requisitarem utilização de drivers, a necessidade de baixas tensões e correntes nos pinos configurados como entrada, e na alimentação, implica na utilização de dispositivos que reduzem os sinais de entrada. Isso por que há situações, como no caso do sinal proveniente do taco-gerador (que pode chegar a 10V), nas quais o sinal a ser lido não é suportado pelo PIC.

Como já mencionado, o microcontrolador PIC 16F877A não suporta tensões superiores a 5,5 V em seus pinos. Por isso foi necessária a adição de um regulador de tensão (componente 7805 do fabricante Texas Instruments), que garante que a tensão na sua saída nunca será superior a 5V, desde que a entrada seja uma tensão de até 35 volts (de acordo com o seu Datasheet), para alimentar o PIC, evitando problemas devido a picos de tensão provenientes da rede ou ainda devido a possíveis ruídos.

Já para os pinos que são configurados como entrada há duas situações. Pinos que recebem sinais provenientes de dispositivos com saída que não está na faixa de 0 a 5V (como o taco-gerador já mencionado) necessitam do uso de amplificadores, com ganho menor que um, ou divisores de tensão para que a amplitude dos sinais seja reduzida e passe a pertencer à faixa adequada.

Além disso, de acordo com o manual do microcontrolador (Microchip PIC 16F87X Data sheet), recomenda-se, nas portas de conversão analógico/digital, uma impedância de fonte de, no máximo, 10 KΩ. Essa impedância máxima é necessária devido ao efeito do “Sample and hold”. Este determina uma impedância máxima para carregar o capacitor ser dentro de um intervalo de tempo satisfatório para a realização da conversão.

Como foram utilizados “trimpots” e potenciômetros com impedâncias superiores a 10 KΩ, foi necessária a adição de um “buffer de tensão”. O “buffer” é um amplificador operacional não inversor de ganho unitário. Este amplificador serve para conectar um estágio de entrada de alta impedância a uma carga que necessita de baixa impedância de entrada. Seu esquema é mostrado na Figura 7.1.

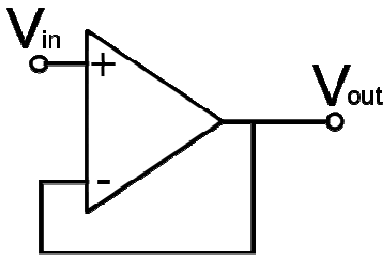


Figura 7.1 - Esquema de um amplificador operaicional utilizado como "buffer de tensão"

Fonte: pt.wikipedia.org

No circuito foram aplicados “buffers de tensão” aos potenciômetros que variam os valores de Kp, Kd, Ki, o sinal do “Setpoint” (velocidade que se deseja alcançar) e à saída do “trimpot”, uma vez que todos apresentam impedâncias superiores ou muito próximas à especificada como máxima (Microchip PIC 16F87X Data sheet). Para isso, foram utilizados dois CI’s (Circuitos Integrados) LM324N (fabricante ST microeletronics), nos quais cada um apresenta 4 amplificadores operacionais.

7.2 PROGRAMAÇÃO

A programação, como já descrito foi feita na linguagem “C”, utilizando o compilador “CCS C Compiler” e o ambiente Microchip MPLAB IDE (Integrated Development Environment) V8.3. O MPLAB é um software que integra a programação, o compilador e o dispositivo utilizado para gravar o PIC (Microchip PICSTART Plus).

Este capítulo apresenta alguns conceitos utilizados na programação e a lógica básica para o seu desenvolvimento. Entretanto, o código utilizado está apresentado em anexo para que se possa obter total entendimento das técnicas utilizadas.

O compilador necessita de que, antes de qualquer instrução de programa, sejam dadas instruções (chamadas diretivas) a ele para configurar sua atuação. Essas são descritas com um caracter “#” antes da instrução.

A programação de um sistema embarcado é feita de maneira que o microcontrolador nunca pare de efetuar as instruções a que se propõe no código, portanto as instruções necessárias à atuação em tempo real devem estar contidas dentro de um “Loop infinito”.

Na linguagem “C” utiliza-se uma função denominada MAIN, que é sempre a primeira a ser feita pelo compilador e deve ser adicionada no fim do código. É nessa função que se fazem configurações das portas a serem utilizadas, as

inicializações das variáveis globais, e na qual são configurados os parâmetros iniciais como definições de entradas A/D, saída PWM e a sua respectiva freqüência. Como de costume, nela foi inserido o “Loop infinito” do programa.

Dentro do “Loop infinito” é chamada a função PID que determina (ou realiza a leitura) dos valores das constantes de controle Kp, Kd, e Ki e efetua a leitura dos sinais do taco-gerador e do “Setpoint”. A mesma função é responsável pelo cálculo do erro, da sua derivada e da sua integral, para serem utilizados na equação que determina a saída do PID.

A saída dessa função é a representação da forma como o microcontrolador deve atuar no motor. Sendo assim, o ciclo ativo do sinal PWM será proporcional a essa saída.

Após a função PID, são chamadas as funções que servem para ativar o mostrador de 7 Dígitos, programado para mostrar a potência que está sendo fornecida ao motor, em função do ciclo ativo, ou seja, 0 a 100%.

Foi implementado um programa que possibilita a aquisição dos parâmetros em tempo real. Entretanto, por limitações de uso do laboratório, não foi possível realizar os testes necessários à validação do funcionamento. Apesar disso,pode-se afirmar que o programa realiza a aquisição dos parâmetros e atua, de acordo com o esperado.

Acredita-se que a grande dificuldade de se obter um controlador eficaz utilizando essa variação de parâmetros em tempo real seja especificar um intervalo no qual as constantes devem estar situadas, dado que é sabido que uma pequena variação nas constantes pode trazer alterações significativas na resposta do sistema a um dado sinal.

Além disso, as constantes ideais não possuem “valores-padrão”, podendo ser qualquer número, ou estar dentro de qualquer intervalo. Como a resolução da porta A/D é de apenas 10 bits, para abranger intervalos maiores, é necessário abrir mão da resolução para valores intermediários.

7.3 AQUISIÇÃO DE DADOS

A aquisição dos dados necessários para a obtenção da função de transferência da planta foi realizada no Laboratório de Ensino de Controle (LECO), através do kit didático DC Servo Trainer ED-4400B, composto dos módulos:

- U-156B – Fonte de alimentação CC (± 15V 0,2A e alimentação do motor)
- U-154B – Driver amplificador de tensão (10 watts)
- U-161B – Servo Motor: 12V; 4,5W
- Taco Gerador: Aproximadamente 3Vp-p/4000rpm
- U-155B – Unidade taco amplificadora
- U-159B – Tacômetro (4000rpm)

Montado conforme mostrado na Figura 7.2.

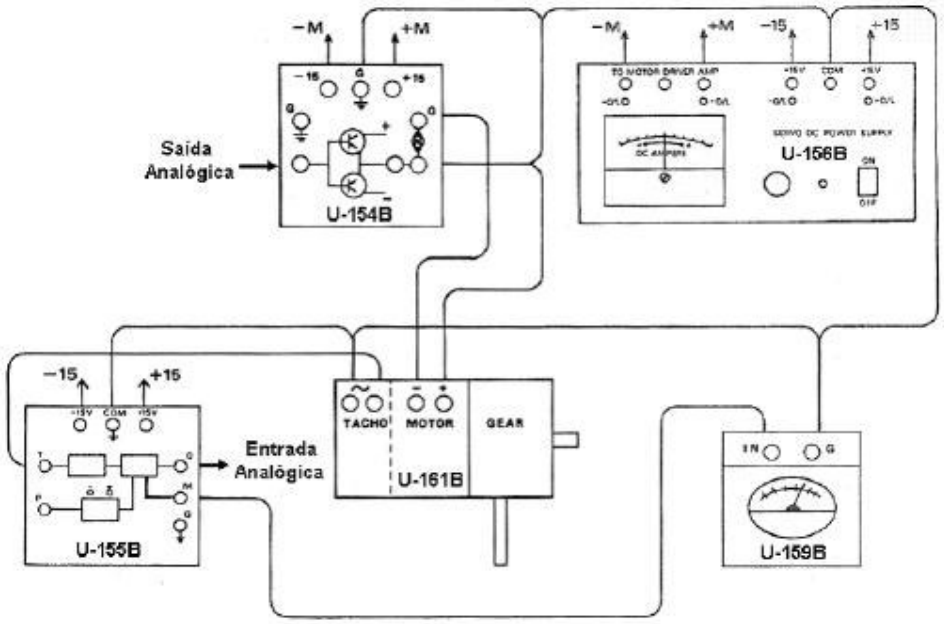


Figura 7.2 - Esquemático do kit educativo na montagem que foi usada nesse projeto

Fonte: Teixeira (2008)

O módulo construído foi conectado de forma que o sinal de saída do taco é enviado diretamente ao PIC, antes de passar pelo amplificador (U-155B), que é usado apenas para possibilitar a medição da velocidade em RPM pelo módulo U-159B. A saída de PWM gerada pelo PIC é conectada de maneira que a tensão fornecida pela fonte de alimentação é liberada para o motor apenas durante o ciclo ativo.

Com esse arranjo foi possível levantar as curvas de tensão de saída e velocidade em RPM x duty cycle, mostradas na Figura 7.3 que foram usadas para realizar conversões necessárias à programação do PID.

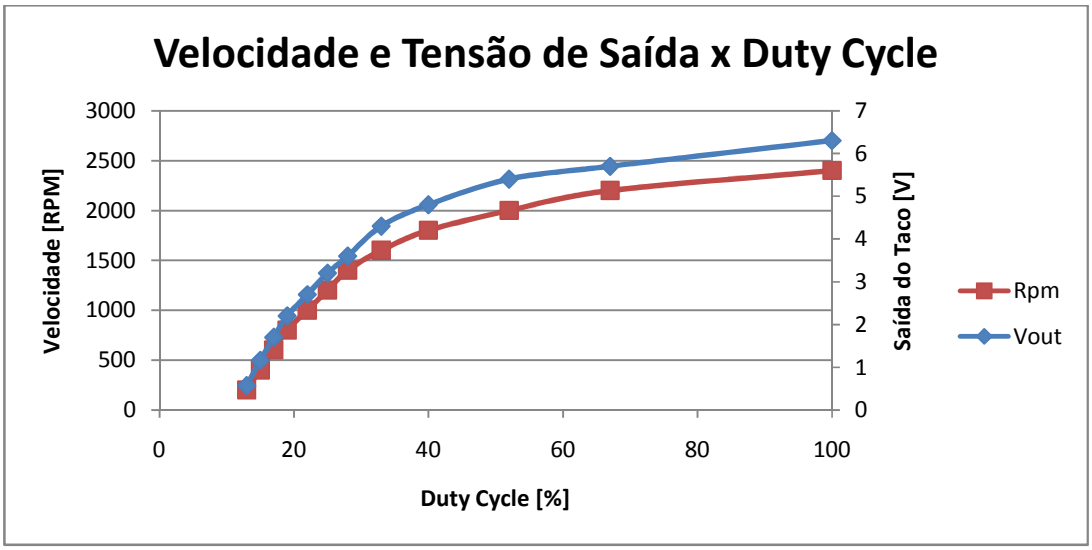


Figura 7.3 – Curva de Velocidade e Tensão x Duty Cycle

Além disso, com a utilização da placa de aquisição NUDAQ PCI-9112 e ferramentas em MatLab, foi feito a aquisição da resposta da planta a uma entrada em degrau de 5V, com o duty cycle de 100%, mostrada na Figura 7.4.

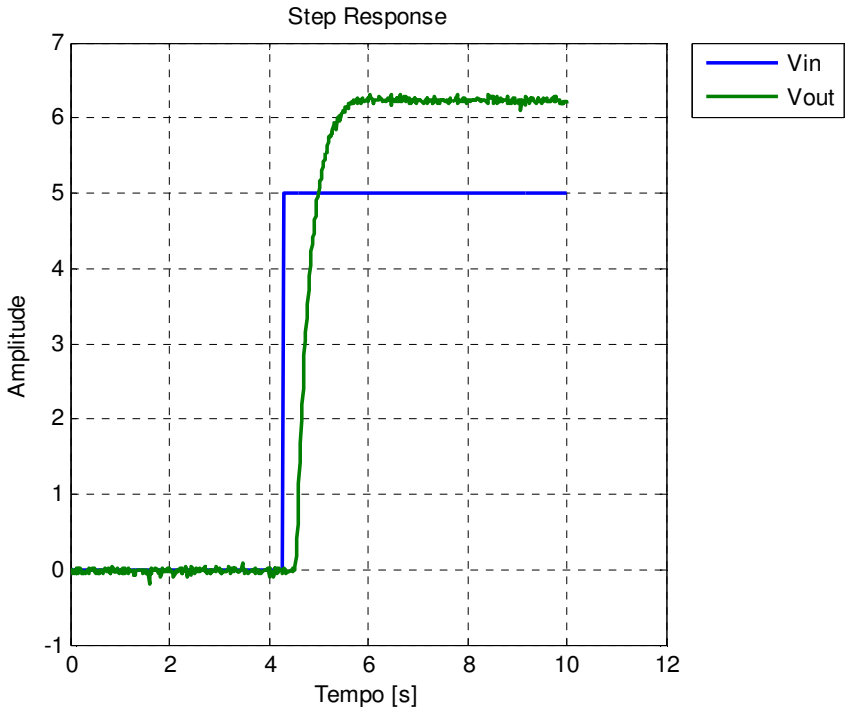


Figura 7.4 - Resposta da planta a um degrau de 5V

7.4 MODELAGEM DA PLANTA

Para modelar a planta, o primeiro passo foi realizar a conversão da tensão de entrada (de 0 a 5V), que define o setpoint, em um número de 0 a 1023, que representa o valor armazenado na variável de 10 bits que lê esse sinal. Nesse caso, o valor 1023 equivale à tensão de 5V.

O mesmo tipo de conversão foi realizada com o sinal proveniente do taco-gerador. Nesse caso, o sinal varia de 0 a 6,4V, entretanto, como a tensão máxima suportada pelo PIC é de 5V, um divisor de tensão foi colocado entre a saída do taco e a entrada do PIC, de forma que tem-se uma faixa de 0 a 3,2V chegando no PIC. Nesse caso, o valor 1023 corresponde à tensão de 3,2V chegando no PIC e conseqüentemente à tensão de 6,4V na saída do taco, que é o valor máximo que pode-se obter.

A resposta da planta ao degrau de 5V, após a conversão, é mostrada na Figura 7.5.

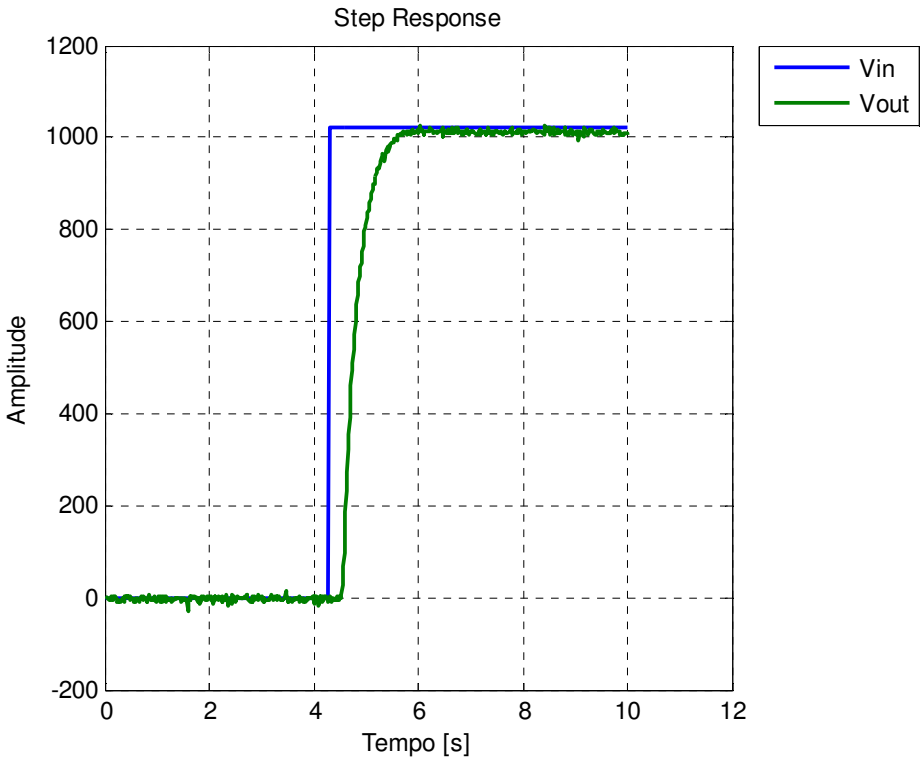


Figura 7.5 - Resposta da planta a um degrau de 5V (valores convertidos)

Como foi explicado anteriormente, o motor de corrente contínua usado para realização dos testes pode ser modelado como um sistema de primeira ordem do tipo:

$$G(s) = \frac{k}{\tau s+1} e^{-td.s} \qquad 7.1$$

Onde:

$k = \frac{\Delta y}{\Delta u}$ é o ganho estacionário;

τ é a constante de tempo;

e t_d é o atraso.

Quando aplica-se um degrau à planta, ela reage da maneira mostrada na Figura 7.6 e então pode-se obter os parâmetros citados acima e assim obter o modelo da planta.

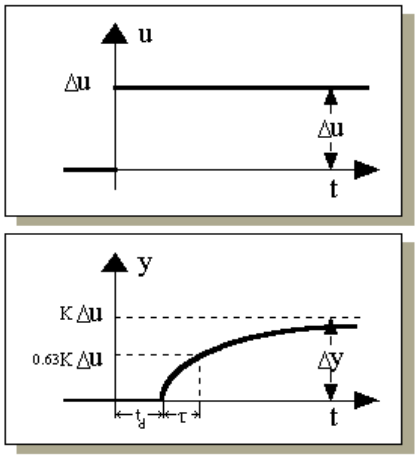


Figura 7.6 - Resposta de um sistema de primeira ordem a uma entrada degrau

Fonte: Teixeira (2008)

A partir da figura 7.5 pode-se obter os valores do tempo de atraso, da constante de tempo e do ganho estacionário. Após alguns ajustes obteve-se o modelo da planta como:

$$G(s) = \frac{0,99}{0,33s+1} e^{-0.15s} \qquad 7.2$$

A comparação entre o modelo encontrado pela aproximação por um sistema de primeira ordem e o resultado real é mostrada na Figura 7.7 onde pode-se notar que o modelo, de fato, corresponde à planta em questão.

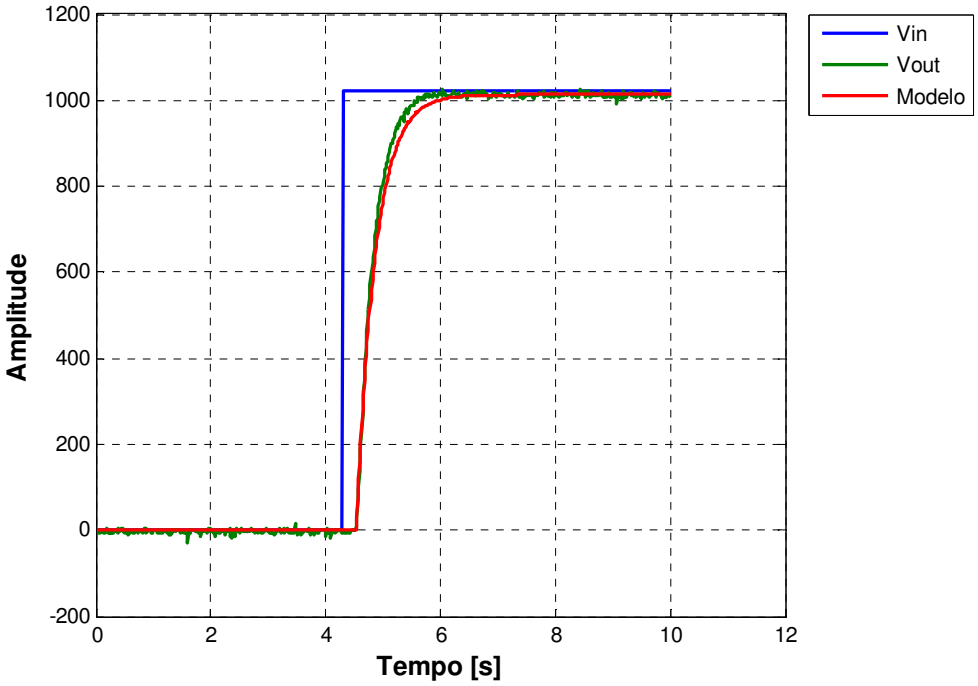


Figura 7.7 - Resposta da planta e do modelo ao degrau de 5V

7.5 PROJETO DO PID E SIMULAÇÕES COMPUTACIONAIS

Verificou-se, como já foi mostrado na Figura 7.3, a não-linearidade na relação entre o duty cycle e a tensão de saída do taco, assim como o duty cycle e a velocidade do motor. A relação entre a saída do taco e a velocidade do motor, entretanto, é linear.

Para adequar a relação exponencial encontrada, essa função foi aproximada para quatro segmentos de retas, que são usadas para corrigir a saída do PID, de maneira a compensar a não-linearidade. Os pontos são mostrados na Figura 7.8 e as retas são mostradas na Figura 7.9.

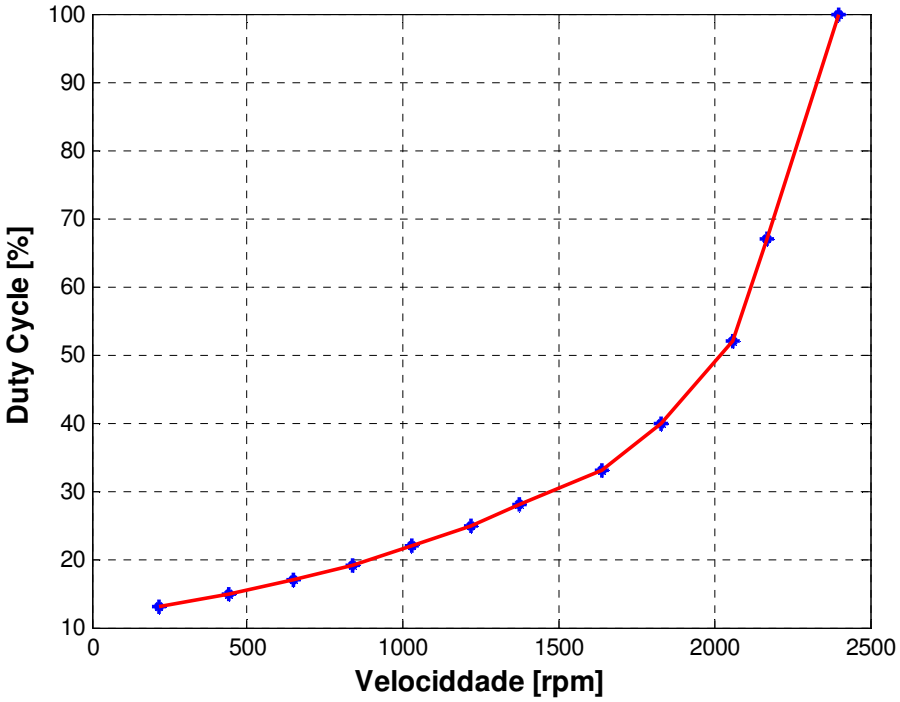


Figura 7.8 - Curva duty cycle x tensão de entrada

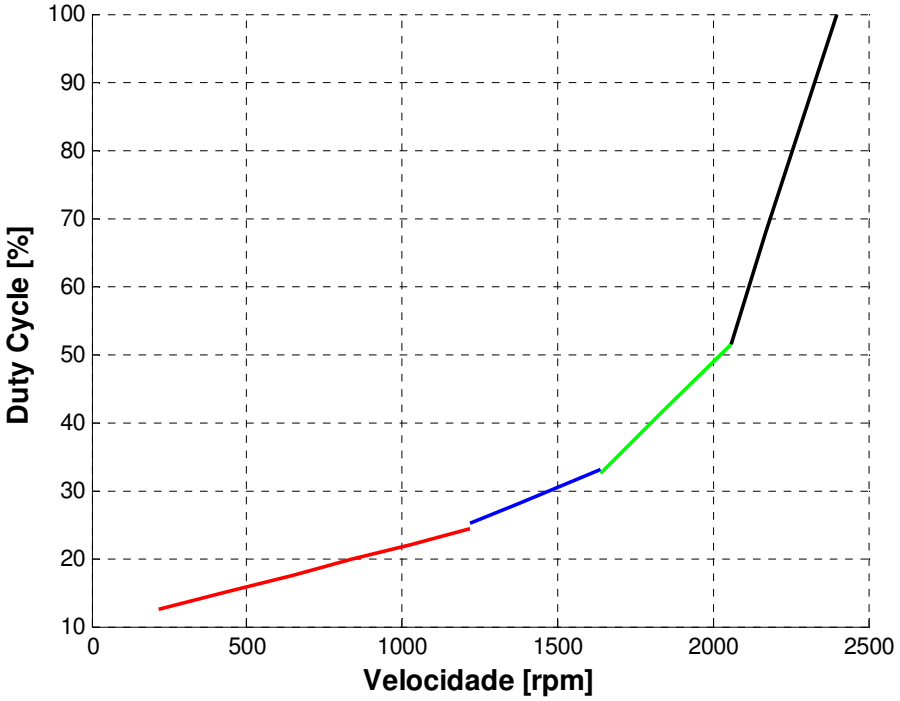


Figura 7.9 - Retas usadas para aproximação

A faixa de utilização de cada reta e seus coeficientes estão na Tabela 7.1.

Tabela 7.1 - Retas de aproximação

	Velocidade [rpm]	Referência Interna [0 a 1023]	Coef. Angular	Coef. Linear
Reta 1	até 1218	até 519	0,0279	9,8028
Reta 2	1218 a 1638	519 a 698	0,0447	1,8118
Reta 3	1638 a 2057	698 a 877	0,1069	-42,2527
Reta 4	2057 a 2400	877 a 1023	0,3299	-237,7143

A partir da função de transferência da planta, foram projetados dois controladores PID, um baseado no primeiro método de Ziegler-Nichols e outro projetado no domínio da freqüência.

O primeiro controlador foi projetado pelo primeiro método de Ziegler-Nichols e os parâmetros obtidos para Kp, Kd e Ki são mostrados na Tabela 7.2

Tabela 7.2 - Valores de Kp, Kd e Ki para o primeiro controlador

Kp	Kd	Ki
2,640	0,198	8,800

Os resultados de simulações computacionais para esse primeiro método são todos comentados na Tabela 7.3.

Tabela 7.3 - Resultados para o primeiro método de Ziegler-Nichols

Figuras	Referência	Comentários
7.10 e 7.11	600 rpm	Estabelecendo-se a referência como uma entrada de 600 rpm, tem-se 57% de overshoot, tempo de acomodação de 1,7s, sem saturação do comando de controle.
7.12 e 7.13	1200 rpm	Estabelecendo-se a referência como uma entrada de 1200 rpm, tem-se 45% de overshoot, tempo de acomodação de 1,7s, com saturação do comando de controle até 0,4s.
7.14 e 7.15	1760 rpm	Estabelecendo-se a referência como uma entrada de 1760 rpm, tem-se 31% de overshoot, tempo de acomodação de 2,2s, com saturação do comando de controle até 1,1s.
7.16 e 7.17	2356 rpm	Estabelecendo-se a referência como uma entrada de 2356 rpm, tem-se um tempo de acomodação de 1,5s com saturação do comando de controle e sem overshoot.

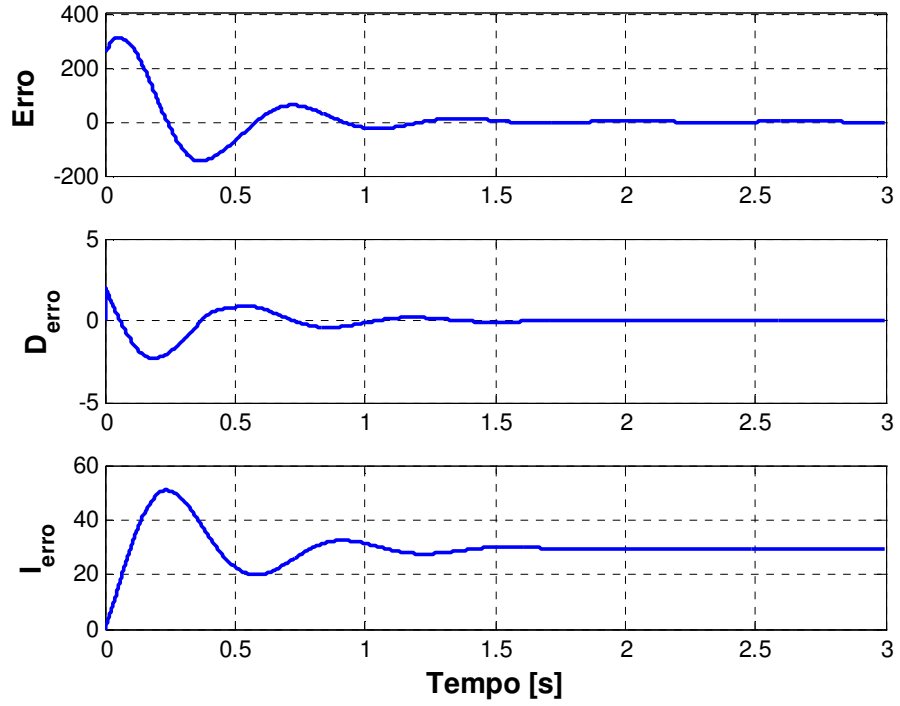


Figura 7.10 - Erro, derivada do erro e integral do erro para referência 600rpm - PID 1

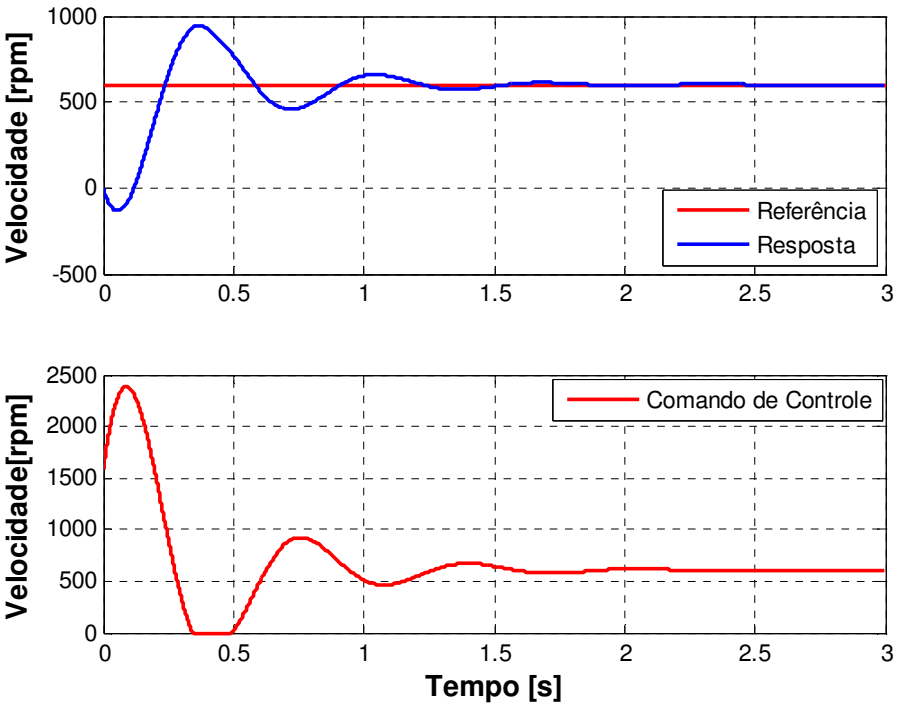


Figura 7.11 - Resposta do sistema e comando de controle para referência 600rpm - PID 1

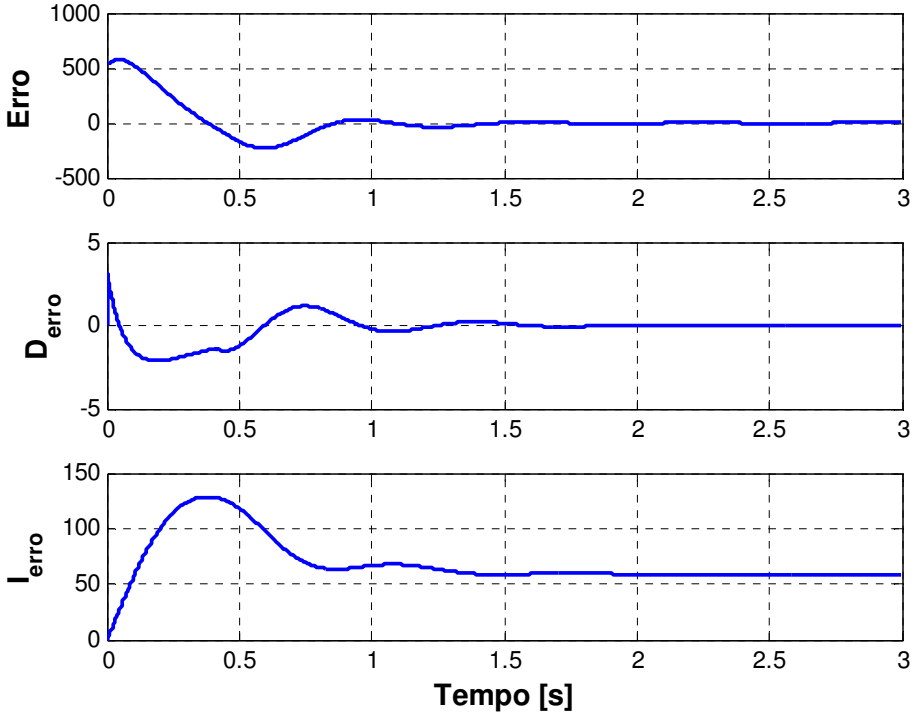


Figura 7.12 - Erro, derivada do erro e integral do erro para referência 1200rpm - PID 1

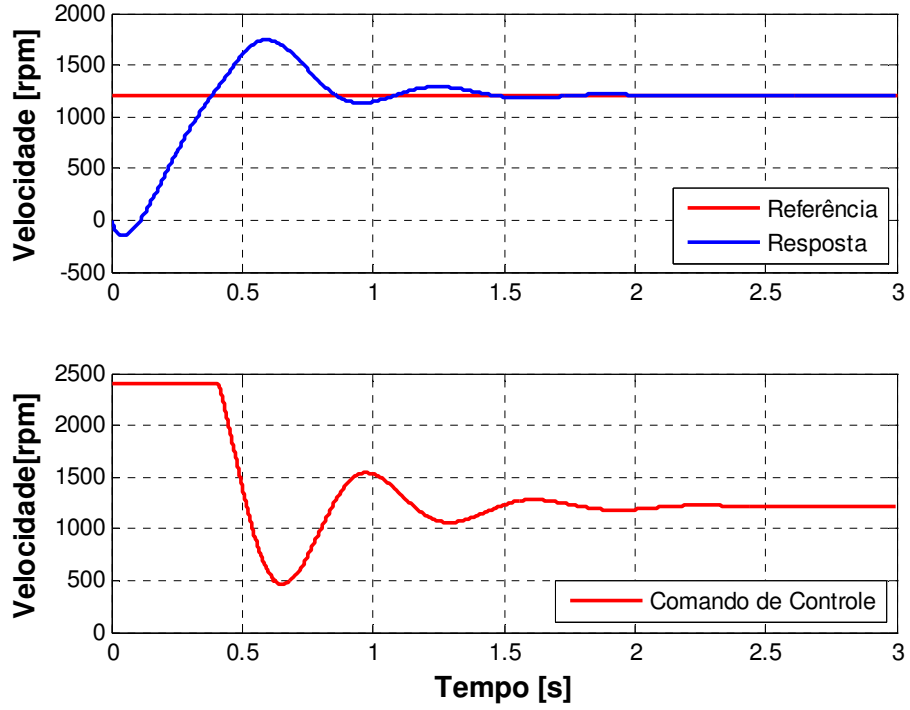


Figura 7.13 - Resposta do sistema e comando de controle para referência 1200rpm - PID 1

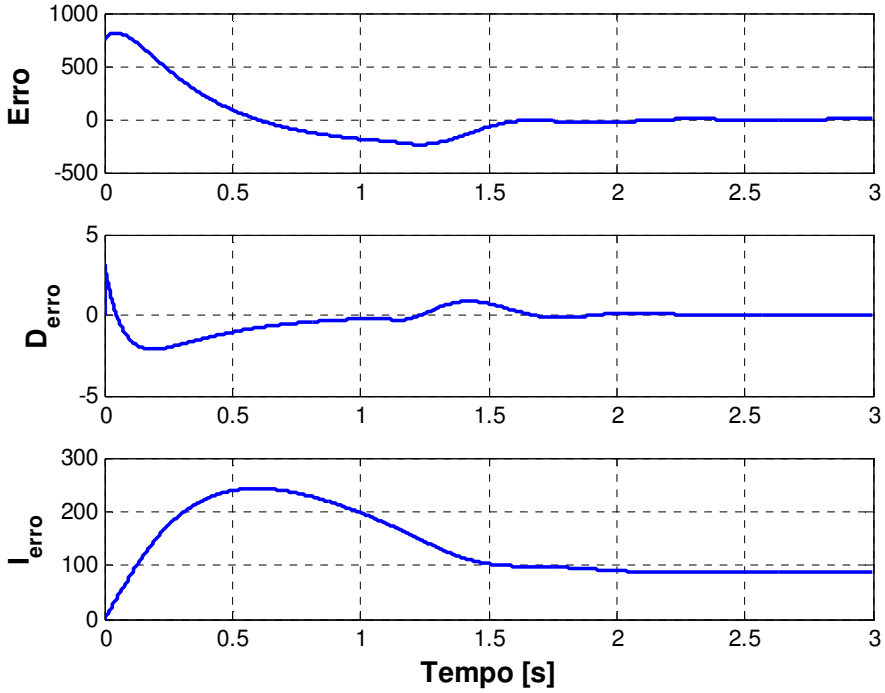
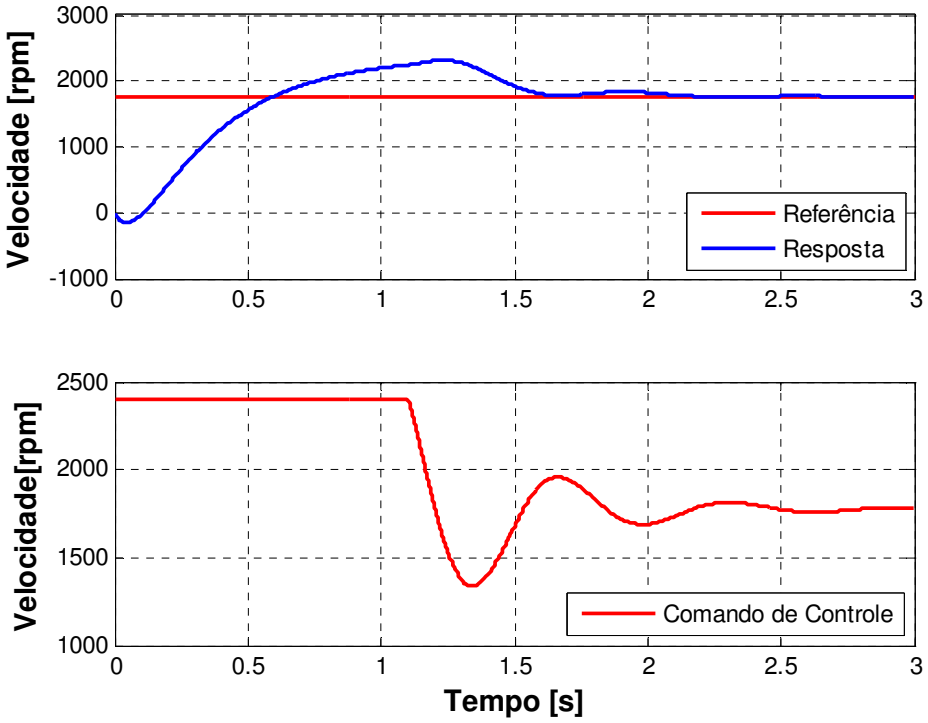


Figura 7.14 - Erro, derivada do erro e integral do erro para referência 1760rpm - PID 1



7.15 - Resposta do sistema e comando de controle para referência 1760rpm - PID 1

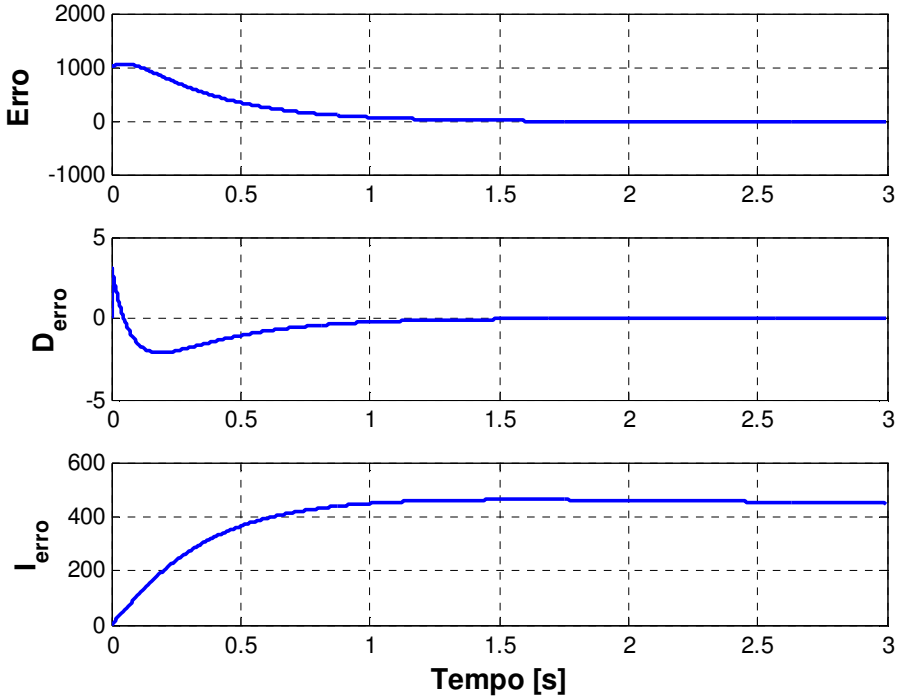


Figura 7.16 - Erro, derivada do erro e integral do erro para referência 2356rpm - PID 1

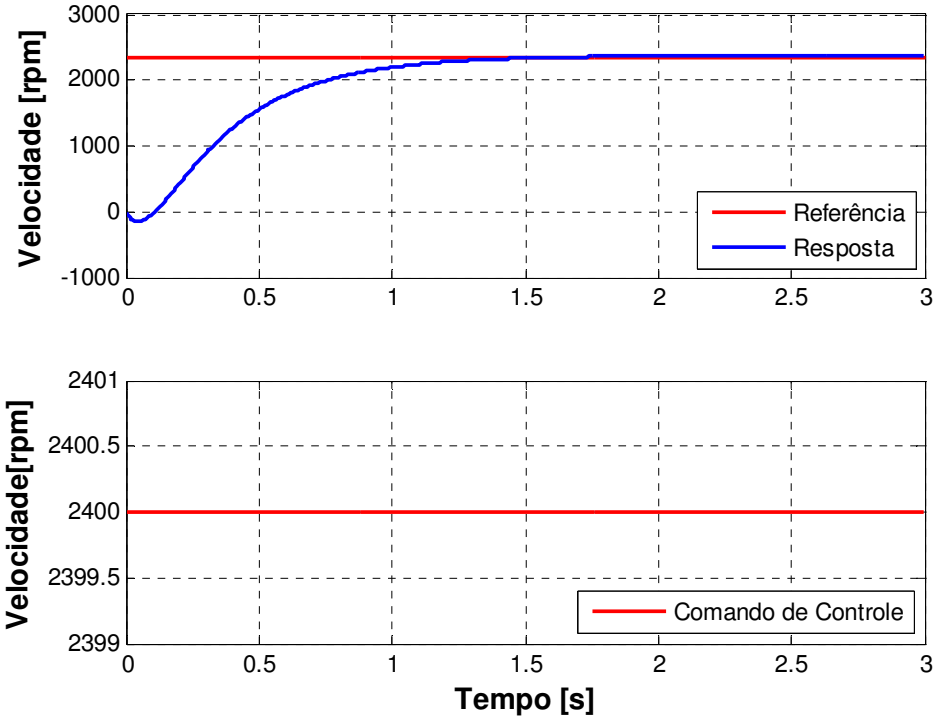


Figura 7.17 -Resposta do sistema e comando de controle para referência 2356rpm - PID 1

O segundo controlador, PID 2, foi projetado no domínio da freqüência. O sistema compensado apresentou uma margem de ganho de 5dB e margem de fase de 49°. Salienta-se que margem de fase (MF) é o atraso de fase adicional que ocorre na freqüência de cruzamento do ganho | G(jω) | = 1. Essa margem é necessária para que o sistema atinja o limiar de instabilidade e seu valor é $\gamma = 180 + \emptyset$. Se $\emptyset > 0$ margem de fase é positivo e se $\emptyset < 0$ margem de fase é negativo. Margem de ganho é o valor do módulo | G(jω) | na freqüência em que o ângulo de fase é -180°. O controlador PID 2 projetado resultou nos parâmetros Kp, Kd e Ki que estão mostrados na Tabela 7.4.

Tabela 7.4 - Valores de Kp, Kd e Ki para o segundo controlador

Kp	Kd	Ki
0,7727	0,0569	2,5790

Os resultados de simulações computacionais para esse segundo método são todos comentados na Tabela 7.5.

Tabela 7.5 - Resultados para o método dois: Mg = 5dB e MF = 49°

Figuras	Referência	Comentários
7.18 e 7.19	600 rpm	Estabelecendo-se a referência como uma entrada de 600 rpm, tem-se um tempo de acomodação de 0,95s, sem saturação do comando de controle e sem overshoot.
7.20 e 7.21	1200 rpm	Estabelecendo-se a referência como uma entrada de 1200 rpm, tem-se um tempo de acomodação de 0,95s, sem saturação do comando de controle e sem overshoot.
7.22 e 7.23	1760 rpm	Estabelecendo-se a referência como uma entrada de 1760 rpm, tem-se um tempo de acomodação de 0,95s, sem saturação do comando de controle e sem overshoot.
7.24 e 7.25	2346 rpm	Estabelecendo-se a referência como uma entrada de 2346 rpm, tem-se um tempo de acomodação de 0,95s com saturação do comando de controle a partir de 0,08s e sem overshoot.

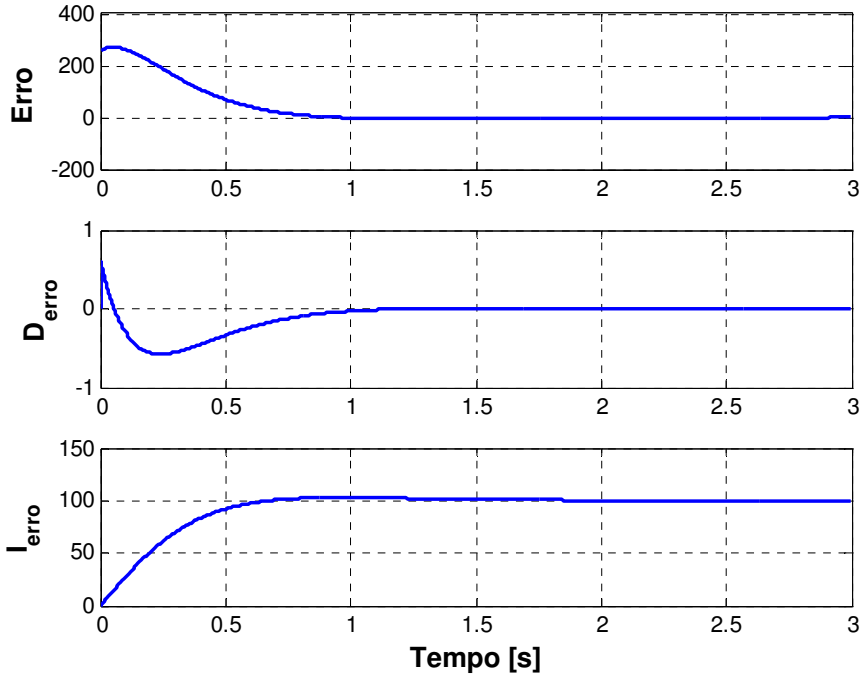


Figura 7.18 - Erro, derivada do erro e integral do erro para referência 600rpm – PID 2

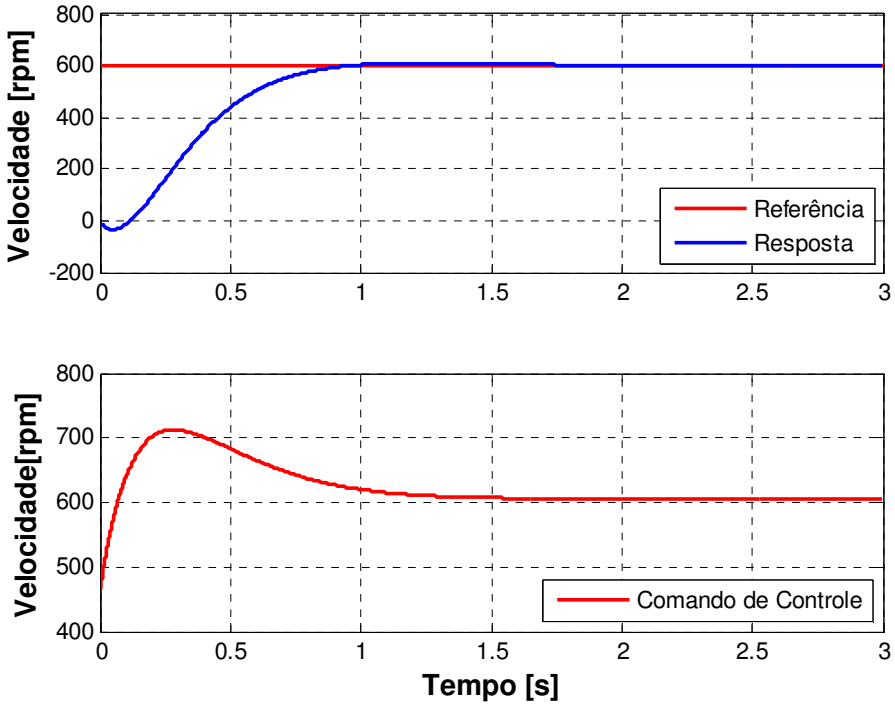


Figura 7.19 - Resposta do sistema e comando de controle para referência 600rpm – PID 2

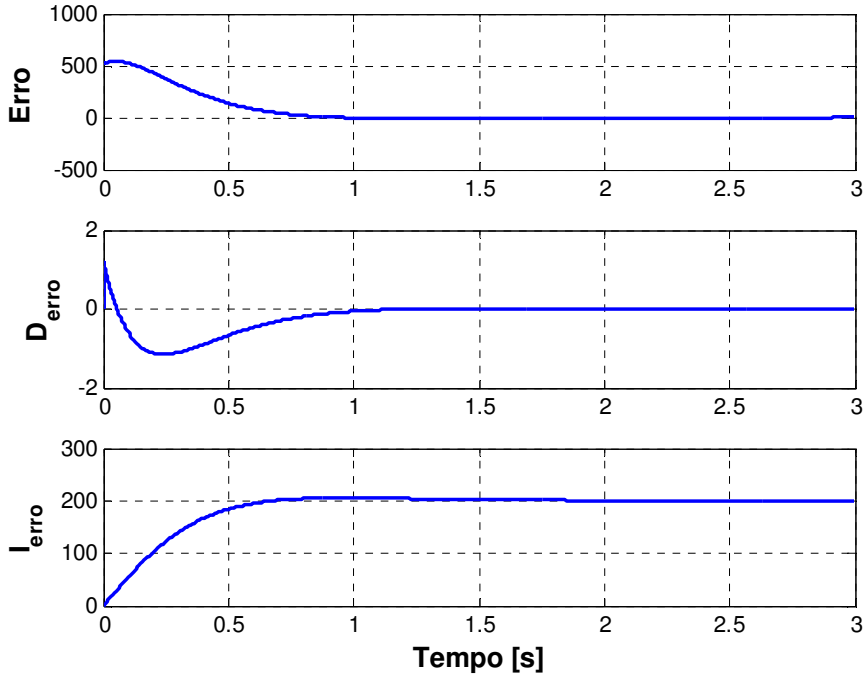


Figura 7.20 - Erro, derivada do erro e integral do erro para referência 1200rpm – PID 2

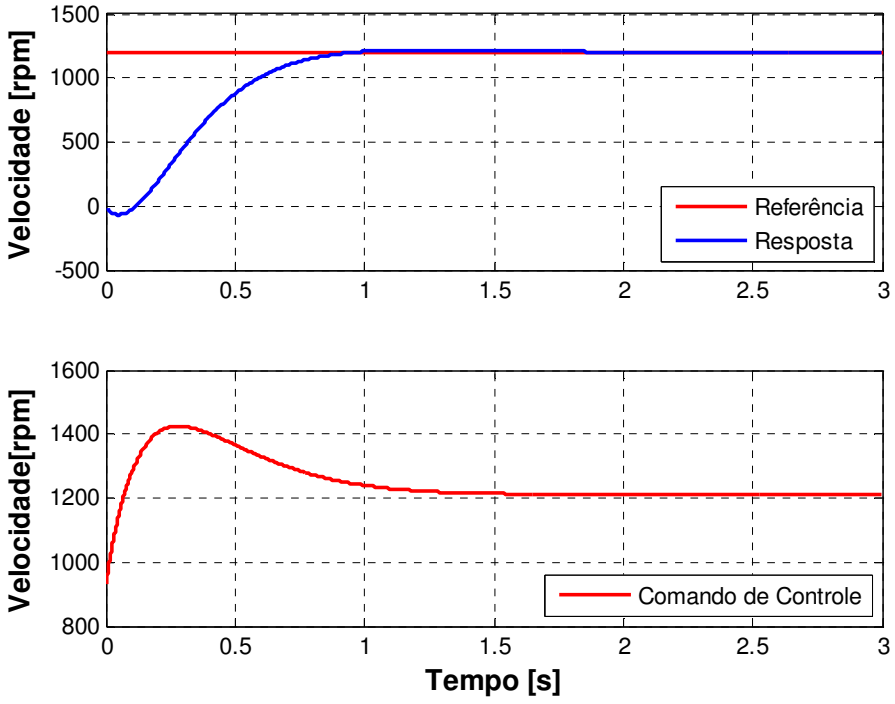


Figura 7.21 - Resposta do sistema e comando de controle para referência 1200rpm – PID 2

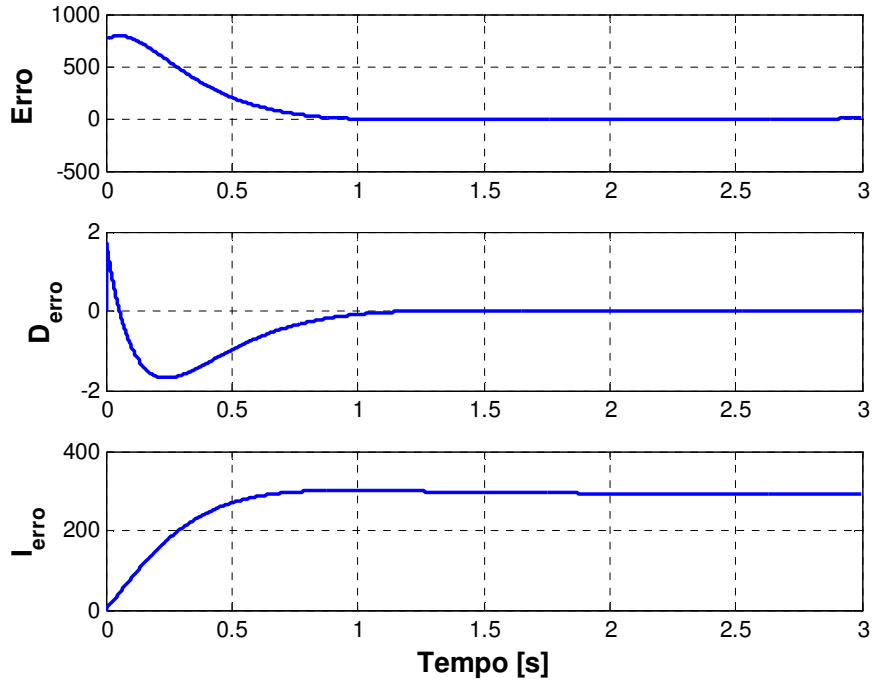


Figura 7.22 - Erro, derivada do erro e integral do erro para referência 1760 – PID 2

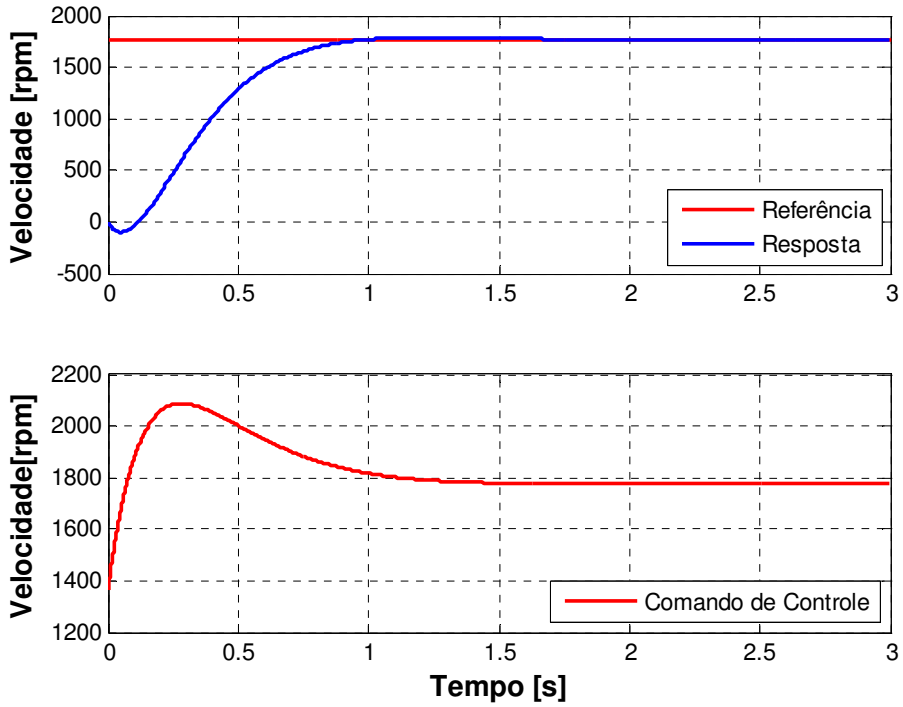


Figura 7.23 - Resposta do sistema e comando de controle para referência 1760 – PID 2

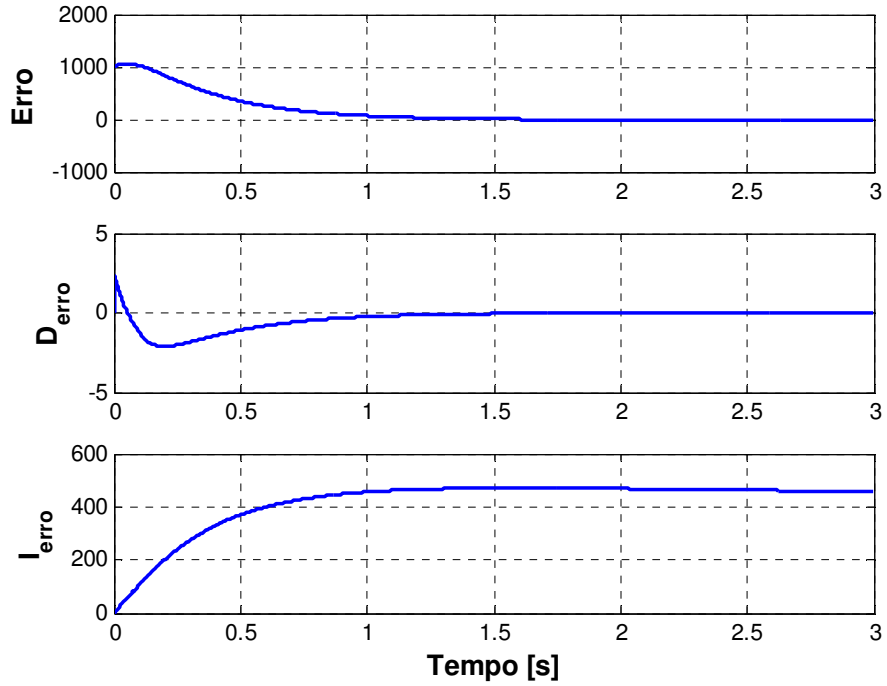


Figura 7.24 - Erro, derivada do erro e integral do erro para referência 2356rpm – PID 2

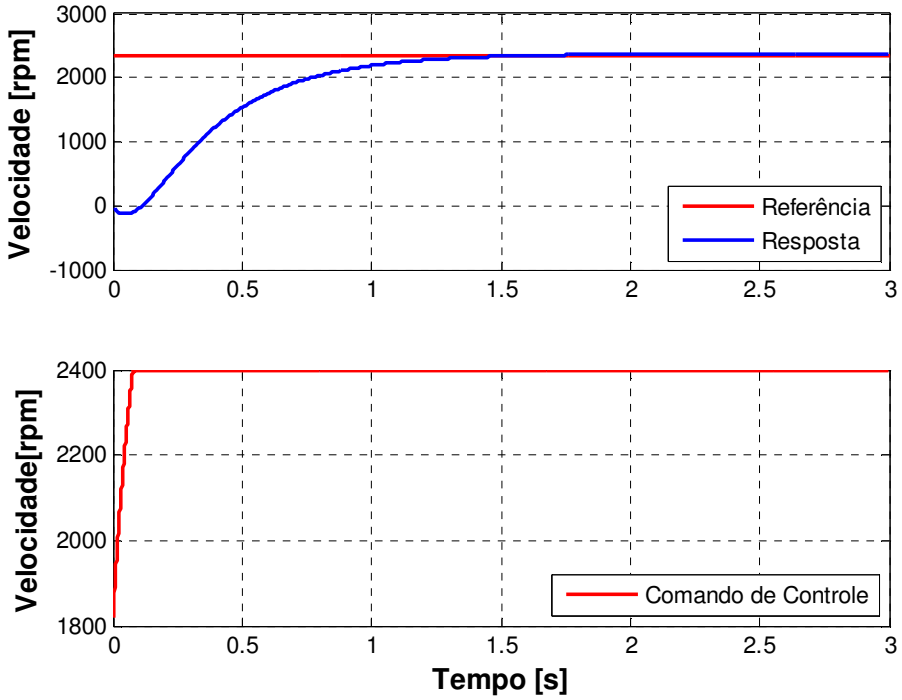


Figura 7.25 - Resposta do sistema e comando de controle para referência 2356rpm – PID 2

7.6 RESULTADOS

Os resultados experimentais foram obtidos no Laboratório de Ensino de Controle (LECO).

Com o controlador PID projetado pelo método de Ziegler-Nichols foram obtidas as respostas mostradas nas figuras 7.26 a 7.30.

Pode-se observar que o controlador funciona. Embora não seja rápido nem preciso, a resposta da planta tende a seguir a referência. Em alguns casos, pode-se observar que existe overshoot, o que está de acordo com a simulação computacional.

Nas figuras 7.26 e 7.27, observa-se que a resposta da planta sobe em dois estágios. Atribui-se esse resultado à aproximação por retas da curva Duty Cycle x RPM, pois a região onde ocorre a transição de um estágio para outro coincide com transição de uma reta para outra.

O erro de offset não era esperado, entretanto, devido ao fato de não haver possibilidade de ajustar o PID após o teste na planta, esse erro é aceitável.

Na figura 7.30, a resposta não segue o primeiro degrau imediatamente devido à inércia do motor

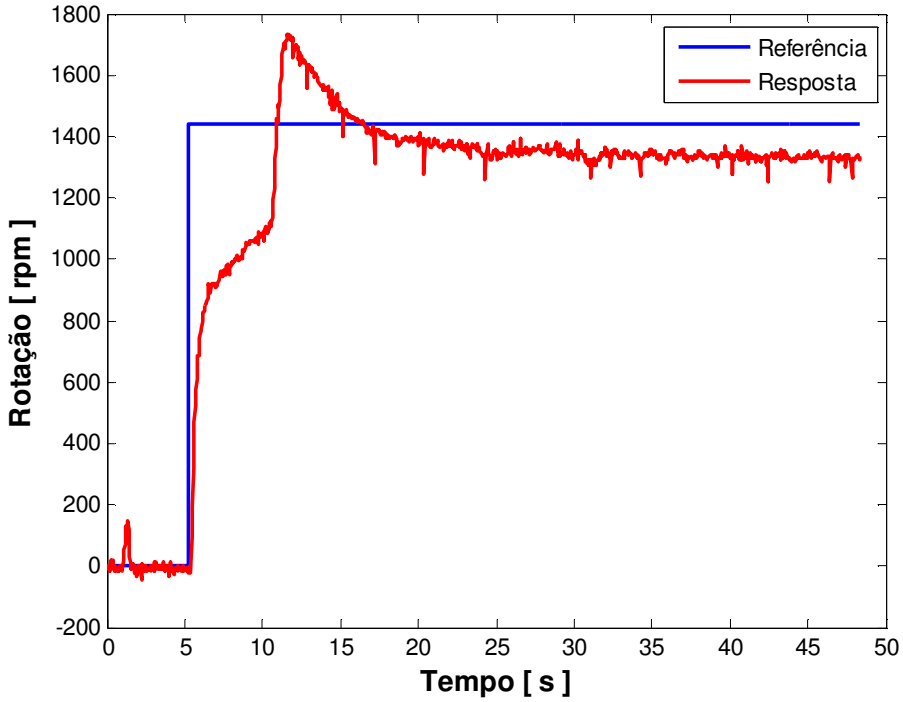


Figura 7.26 - Resposta ao degrau - PID 1

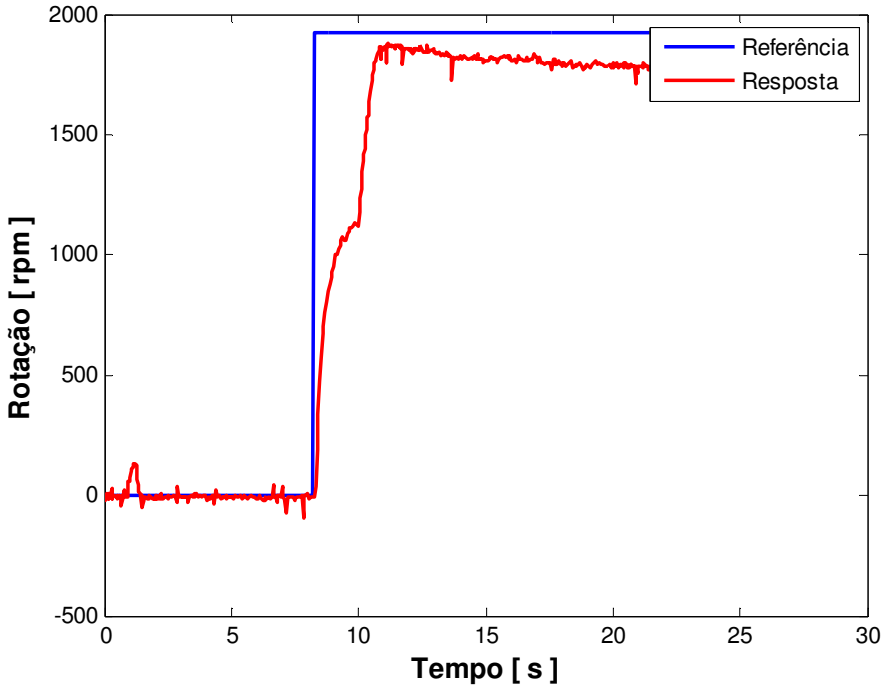


Figura 7.27 - Resposta ao degrau - PID1

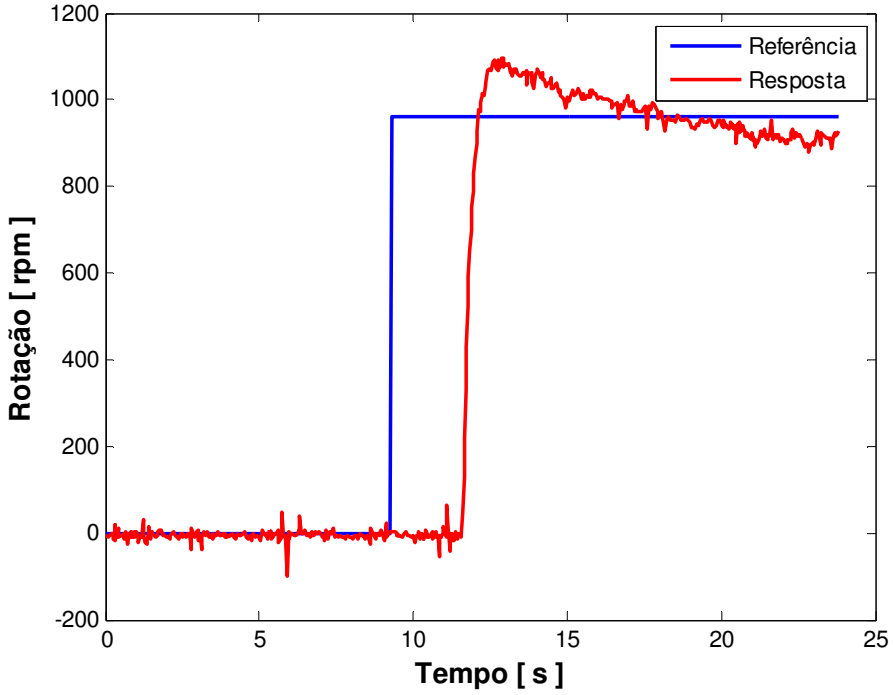


Figura 7.28 -- Resposta ao degrau - PID1

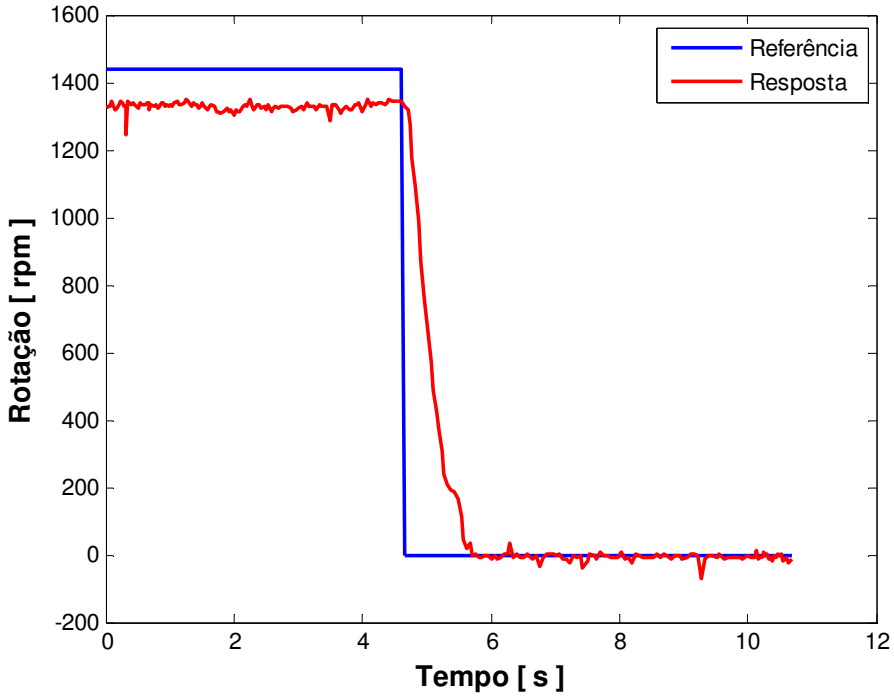


Figura 7.29 - Resposta ao degrau - PID1

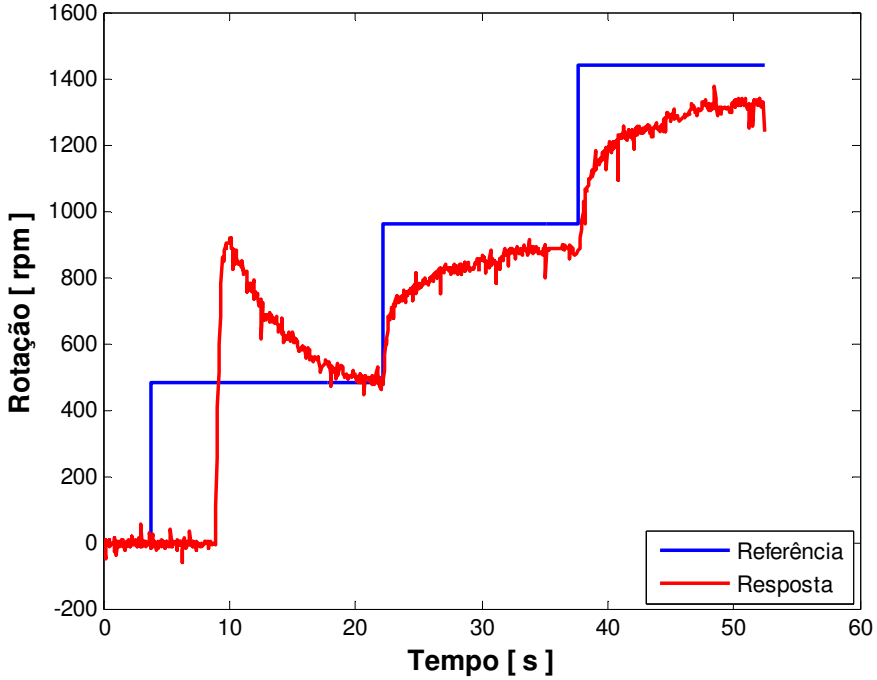


Figura 7.30 - Resposta ao degrau - PID1

Com o segundo controlador, projetado por margem de fase e de ganho obteve-se as respostas mostradas nas figuras 7.31 a 7.34.

Esse controlador apresentou respostas mais rápidas em relação ao controlador 1, e sem overshoot, como era esperado de acordo com a simulação computacional. Entretanto, o erro de offset também aparece na resposta desse controlador e como foi citado anteriormente, é aceitável pois não foi possível realizar outros ajustes do PID após o teste.

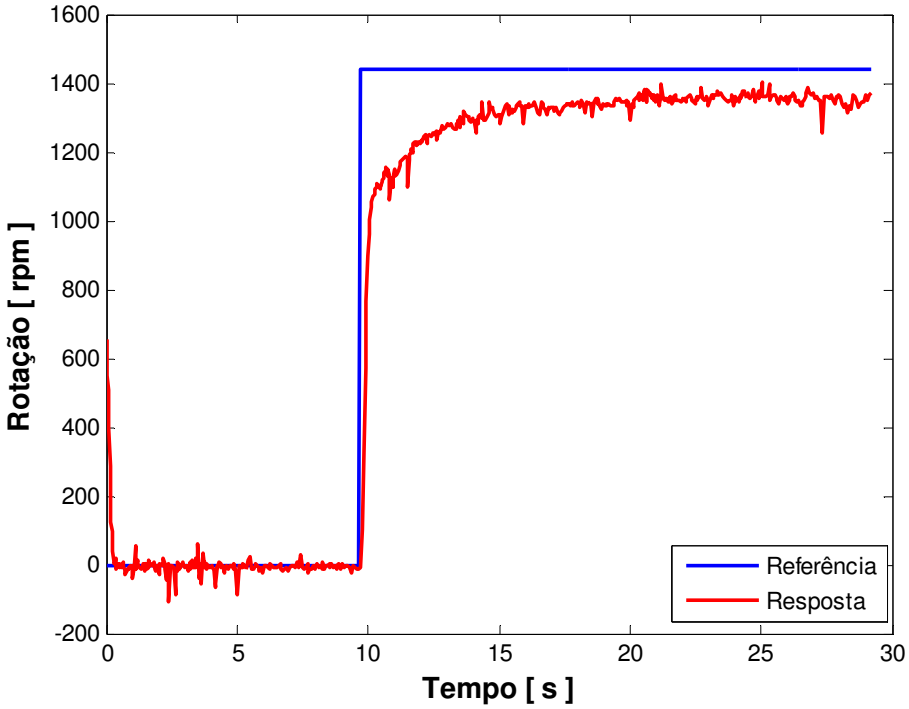


Figura 7.31 - Resposta ao degrau - PID2

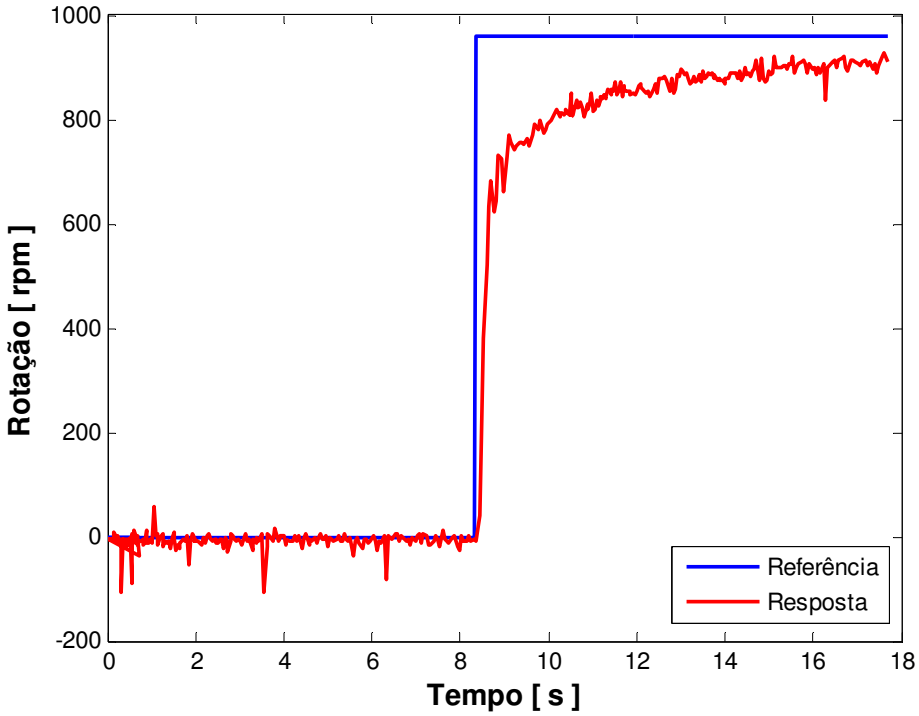


Figura 7.32 - Resposta ao degrau - PID2

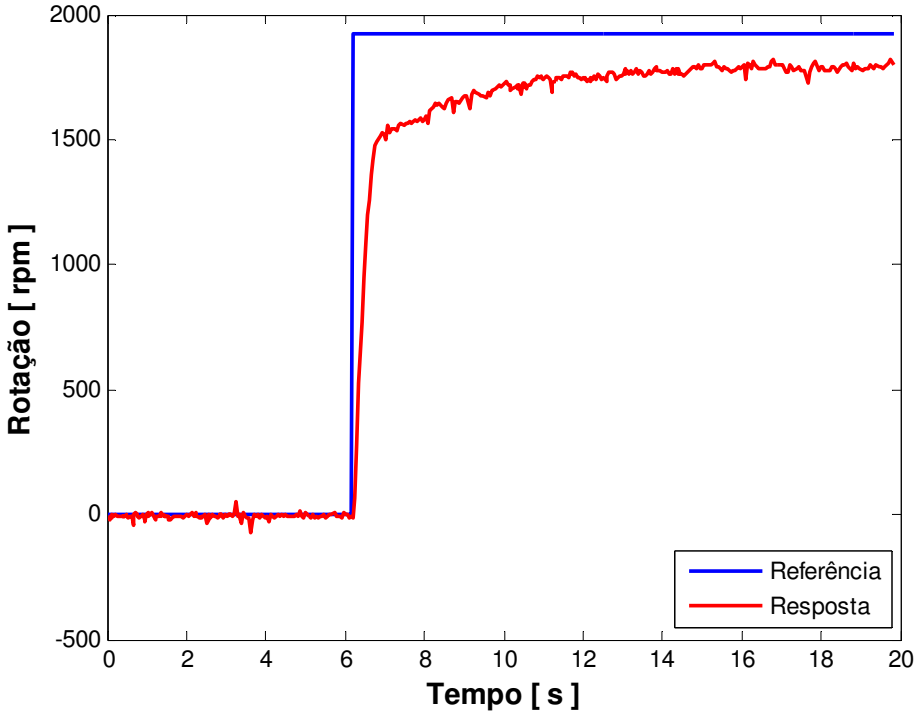


Figura 7.33 - Resposta ao degrau - PID2

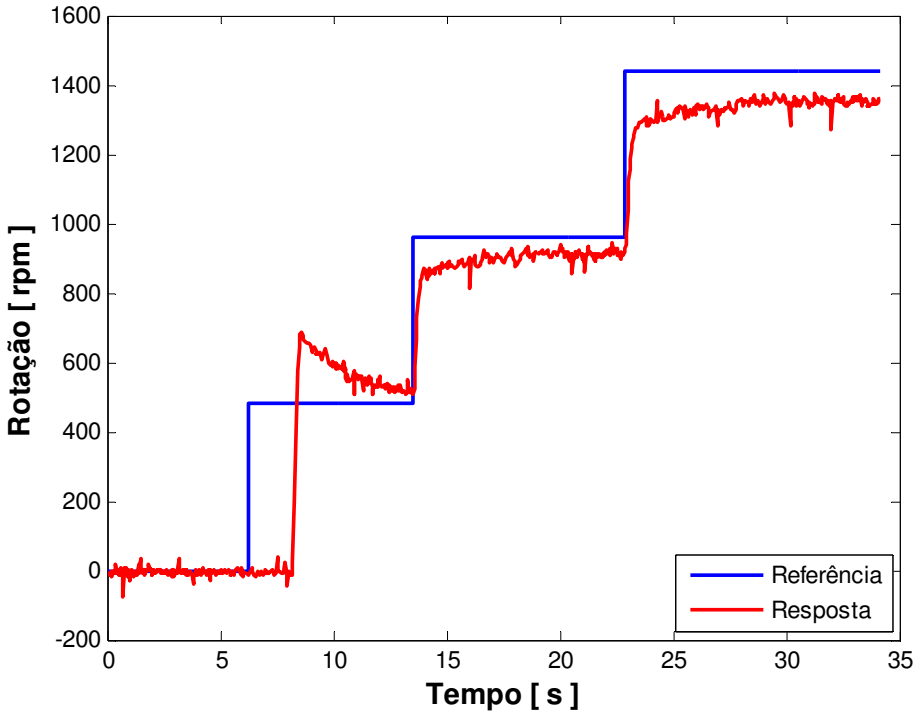


Figura 7.34 - Resposta ao degrau - PID2

8 CONCLUSÃO E FUTURAS PERSPECTIVAS

O trabalho mostrou a possibilidade da aplicação de microcontroladores PIC no controle de velocidade de motores DC. Mais do que isso, foi visto que essa área do conhecimento pode, e deve, ser mais explorada dentro do curso de Engenharia Mecânica, visto que a sua aplicabilidade ultrapassa as limitações desse trabalho.

O controle proposto,via modulação por largura de pulso (PWM), se mostrou viável tecnicamente e eficaz, apesar se perceber que este gera algumas não-linearizações quando analisamos a velocidade de saída em função do duty cycle. Foi verificado que o controle PWM pode ser aplicado, aproveitando-se as suas características positivas, como alteração da velocidade sem alteração de torque e dissipação excessiva de potência.

A programação em “C” se demonstrou muito conveniente, pois foi de fácil assimilação devido ao conhecimento prévio dos autores em programação em outras linguagens de alto nível.

Ainda foram verificadas técnicas para projeto de PID’s, ou seja, determinação de suas constantes. As técnicas verificadas foram os métodos de Ziegler-Nichols, o método de margem de ganho e margem de fase, utilizando o Matlab.

O projeto ainda deixa espaço para implementação de melhorias. Em pesquisas futuras espera-se que se dê continuidade à tentativa de implantar o sistema de mudança de parâmetros de controle em tempo real. Embora o circuito elétrico e o código já se apresentem com essa possibilidade, não se conseguiu o controle com essa variação devido tanto à sensibilidade quanto à escolha da faixa de valores possíveis para Kp, Kd e Ki.

Espera-se ainda que, com o conhecimento desenvolvido na utilização de microcontroladores, se dê continuidade à sua aplicação utilizando outras ferramentas extremamente disponíveis nesses CI’s, como interrupções, outros timers, visores de LCD, dispositivos de comunicação via portas RS232, dentre outros.

A continuidade das pesquisas com microcontroladores renderá melhorias significativas se forem aplicados métodos de tratamento de interrupções para que o tempo de amostragem seja efetuado de maneira mais constante, gerando bons frutos para o projeto do PID e para aumentar a velocidade e precisão do controle de velocidade.

Há a expectativa de aplicação dessa técnica de controle de velocidade em sistemas embarcados. Um exemplo de campo de atuação dessa técnica seria sua utilização em robôs móveis.

9 REFERÊNCIAS

Albuquerque, P. U., & Bastos, C. G. (Outubro de 1990). Controle de velocidade de motores CC. *Revista Tecnologia* , 32-36.

Aström, K. J., & Wittenmark, B. (1997). *Computer Controlled Systems: theory and design.* Upper Saddle River: Prentice Hall.

Ball, S. (2004). *Analog Interfacing to Embeded Microprocessors.* Burlington: Newness.

Carrilho, A. (2005). *Sensores - Parte 1*. Acesso em 27 de outubro de 2009, disponível em Automação de sistemas e instrumentação industrial: http://www.ime.eb.br/~aecc/Automacao/Sensores_Parte_1.pdfhttp://www.ime.eb.br/~aecc/Automacao/Sensores_Parte_1.pdf

Centinkunt, S. (2008). *Mecatrônica.* Rio de Janeiro, Brasil: LTC - Livros Técnicos e Científicos.

Dogan, I. (2006). *Microcontroller based Applied: Digital Control.* Inglaterra: John Wiley & Sons.

Eletrônica, M. (s.d.). *Mikro Eletrônica*. Acesso em 18 de Outubro de 2009, disponível em Mikro Eletrônica On-line: Introdução aos Microcontroladores: http://www.mikroe.com/pt/product/books/picbook/capitulo1.htm

Franklin, G. F., Powell, J. D., & Workman, M. (1998). *Digital control of dynamic systems.* Menlo Park: Addison Wesley Longman.

Ghirardello, A. (200-?). *Apostila sobre modulação PWM.* Si.

Honda, F. (2006). *Motores de corrente contínua: Guia rápido para uma especificação precisa.* São Paulo.

Kilian, C. T. (2000). *Modern Control Technology: Components and Systems.* Delmat Thomson Learning.

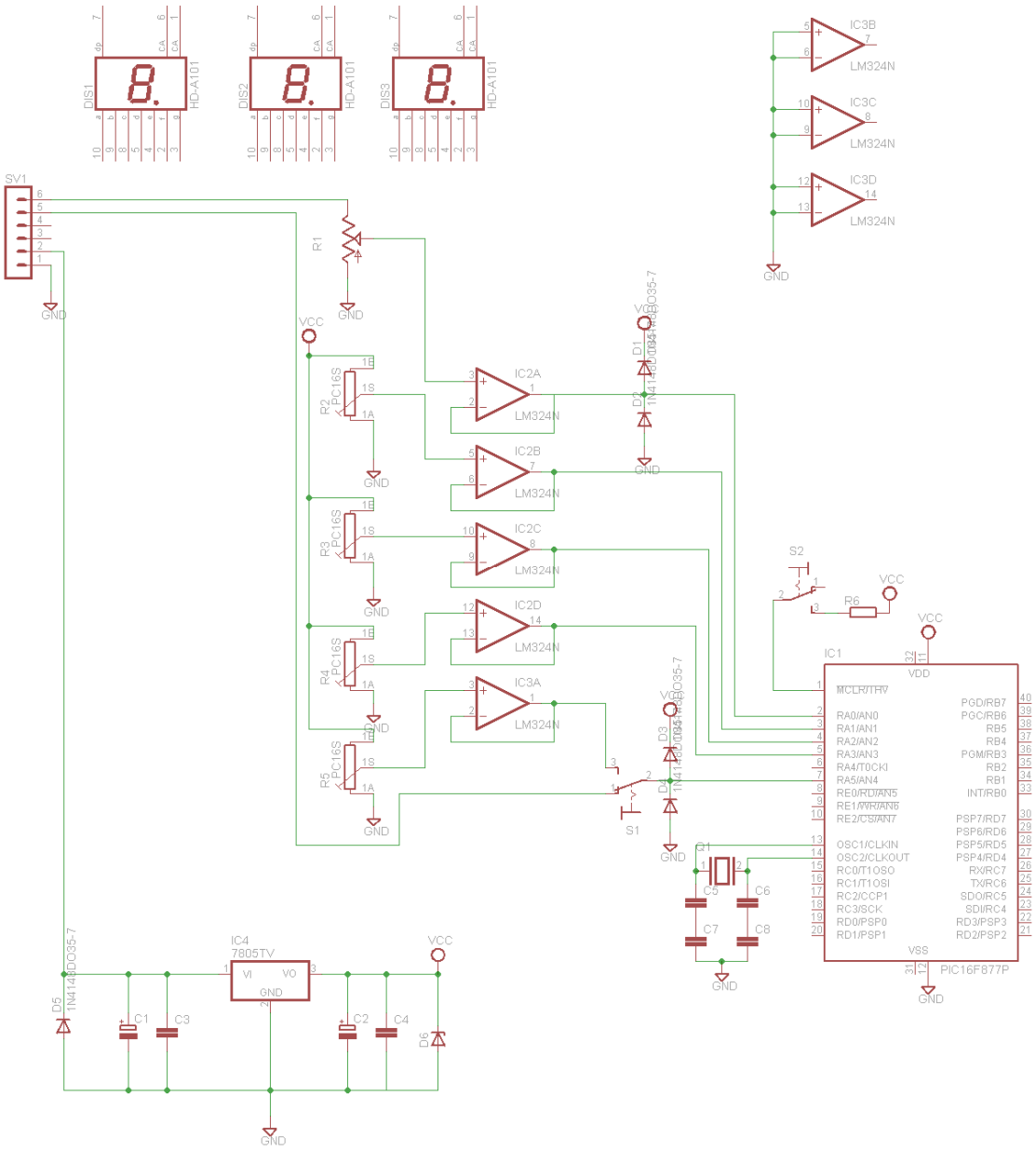
Oliveira, A. L. (1999). *Fundamentos de Controle de Processo.* Vitória: SENAI/CST.

Pereira, F. (2003). *Microcontroladores PIC: Programação em C.* São Paulo: Érica.

Silva, R. A. (2006). *Programando Microcontroladores PIC: Linguagem "C".* São Paulo: Ensino Profissional.

Teixeira, H. T. (Agosto de 2008). DESENVOLVIMENTO DE UM AMBIENTE DE CONTROLE E MONITORAMENTO EM TEMPO REAL USANDO O MATLAB E A PLACA DE AQUISIÇÃO NUDAQ PCI-9112. Vitória, ES, Brasil.

10 ANEXO A – ESQUEMÁTICO DO CIRCUITO



- Observação: Foram ocultados os conectores do visor de 7 dígitos por motivos estéticos.

11 ANEXO B – CÓDIGO-FONTE

```
//************************************************************************
//                CONTROLE PID PARA MOTOR CC COM SAÍDA PWM          *
//                                                                  *
// Este programa realiza a leitura de entradas analógicas, as portas RA0, RA1 *
// RA2, RA3 e RA5, que são definidas pelos potenciômetros do módulo PID.   *
// A partir daí, é definida a saída, que seta o valor do ciclo ativo do PWM,       *
// pino RC2. O valor do ciclo ativo do PWM é mostrado em visor de 7 dígitos,   *
// no formato  do percentual de potência fornecida ao motor                *
//                                                                  *
// Autores: Felipe Padilha Criscuoli                                *
//          Beatriz Leal Miranda                                    *
//                                                                  *
// Data: Nov/2009                                                   *
//************************************************************************
#include <pic16f877a.h>              // Diretivas de compilação:

//#include <16f876.h>               // Diretivas de compilação: PIC 16f877A
#device adc=10                      // Define conversão analógico-digital
                                    // com 10 bits (valor de 0 a 1023)
#fusesXT,NOWDT,PROTECT,PUT,NOLVP //Cristal 4 MHz, sem watchdog, etc.
#use delay(clock=4000000)           // Clock para as rotinas de delay
```

```
//*****************************************************************************
// Definição de Entradas e Saídas                                             *
//*****************************************************************************
//
//
// A pinagem do PIC 16f877A:
//
//                      _____     _____
//      MCLR - 01 --|        |__|        |-- 40 - RB7
//       RA0 - 02 --|                    |-- 39 - RB6
//       RA1 - 03 --|                    |-- 38 - RB5
//       RA2 - 04 --|                    |-- 37 - RB4
//       RA3 - 05 --|                    |-- 36 - RB3
//       RA4 - 06 --|                    |-- 35 - RB2
//       RA5 - 07 --|                    |-- 34 - RB1
//       RE0 - 08 --|                    |-- 33 - RB0
//       RE1 - 09 --|                    |-- 32 - Vdd+
//       RE2 - 10 --|                    |-- 31 - Vss
//      +Vdd - 11 --|                    |-- 30 - RD7
//       Vss - 12 --|                    |-- 29 - RD6
//      OSC1 - 13 --|                    |-- 28 - RD5
//      OSC2 - 14 --|                    |-- 27 - RD4
//       RC0 - 15 --|                    |-- 26 - RC7
//       RC1 - 16 --|                    |-- 25 - RC6
//       RC2 - 17 --|                    |-- 24 - RC5
```

```
//      RC3 - 18 --|                |-- 23 - RC4
//      RD0 - 19 --|                |-- 22 - RD3
//      RD1 - 20 --|____________|-- 21 - RD2
//
// Pinos de I/O (Portas A, B, C, D e E):
//
// porta,0 - Entrada analógica - Sinal de entrada do taco-gerador
// porta,1 - Entrada analógica - Kp
// porta,2 - Entrada analógica - Kd
// porta,3 - Entrada analógica - Ki
// porta,4 - Não usado
// porta,5 - Entrada Analógica - Setpoint ou gerador de sinal
// portb,0 - Visor de 7 segmentos
// portb,1 - Visor de 7 segmentos
// portb,2 - Visor de 7 segmentos
// portb,3 - Visor de 7 segmentos
// portb,4 - Visor de 7 segmentos
// portb,5 - Visor de 7 segmentos
// portb,6 - Visor de 7 segmentos
// portb,7 - Visor de 7 segmentos
// portc,0 - Visor de 7 segmentos
// portc,1 - Visor de 7 segmentos
// portc,2 - Saída PWM
// portc,3 - Chave para utilização de malha aberta/fechada
// portc,4 - Visor de 7 segmentos
```

```
// portc,5 - Visor de 7 segmentos
// portc,6 - Visor de 7 segmentos
// portc,7 - Visor de 7 segmentos
// portd,0 - Visor de 7 segmentos
// portd,1 - Visor de 7 segmentos
// portd,2 - Visor de 7 segmentos
// portd,3 - Visor de 7 segmentos
// portd,4 - Visor de 7 segmentos
// portd,5 - Visor de 7 segmentos
// portd,6 - Visor de 7 segmentos
// portd,7 - Visor de 7 segmentos
// porte,0 - Não usado
// porte,1 - Não usado
// porte,2 - Não usado

//*****************************************************************************
// Programa Principal                                                         *
//*****************************************************************************

int uni,cen,dez;
unsigned long duty;
signed long   erro_ant;                          //   Erro[-1]
double   soma_erro;                            //   Soma dos Erros (integral)
```

```
////////////////////////////////////////
        //UNIDADE//
////////////////////////////////////////
void unidade(uni){

        switch (uni)
        {
                case 0:
                        output_D(0b11000000);
                        break;
                case 1:
                        output_D(0b11111001);
                        break;
                case 2:
                        output_D(0b10100100);
                        break;
                case 3:
                        output_D(0b10110000);
                        break;
                case 4:
                        output_D(0b10011001);
                        break;
                case 5:
                        output_D(0b10010010);
                        break;
```

```
                case 6:
                        output_D(0b10000010);
                        break;
                case 7:
                        output_D(0b11111000);
                        break;
                case 8:
                        output_D(0b10000000);
                        break;
                case 9:
                        output_D(0b10010000);
                        break;
        }
}
////////////////////////////////////////
        //CENTENA//
////////////////////////////////////////
void centena(cen){
        switch (cen)
        {
                case 0:
                        output_B(0b11000000);
                        break;
                case 1:
                        output_B(0b11111001);
```

```
        break;
case 2:
        output_B(0b10100100);
        break;
case 3:
        output_B(0b10110000);
        break;

case 4:
        output_B(0b10011001);
        break;
case 5:
        output_B(0b10010010);
        break;
case 6:
        output_B(0b10000010);
        break;
case 7:
        output_B(0b11111000);
        break;
case 8:
        output_B(0b10000000);
        break;
case 9:
        output_B(0b10010000);
```

```
                        break;
                }
}
////////////////////////////////////////
        //DEZENA//
////////////////////////////////////////
void dezena(dez){
                switch (dez)
                {
                        case 0:
                        output_low (PIN_C0);
                        output_low (PIN_C3);
                        output_low (PIN_C4);
                        output_low (PIN_C5);
                        output_high (PIN_C6);
                        output_low (PIN_C7);
                        output_low (PIN_D7);
                                break;
                        case 1:
                        output_high (PIN_C0);
                        output_high (PIN_C3);
                        output_high (PIN_C4);
                        output_high (PIN_C5);
                        output_high (PIN_C6);
                        output_low (PIN_C7);
```

```
output_low (PIN_D7);
        break;
case 2:
output_low (PIN_C0);
output_low (PIN_C3);
output_low (PIN_C4);
output_high (PIN_C5);
output_low (PIN_C6);
output_high (PIN_C7);
output_low (PIN_D7);
        break;
case 3:
output_low (PIN_C0);
output_low (PIN_C3);
output_high (PIN_C4);
output_high (PIN_C5);
output_low (PIN_C6);
output_low (PIN_C7);
output_low (PIN_D7);
        break;
case 4:
output_high (PIN_C0);
output_high (PIN_C3);
output_high (PIN_C4);
output_low (PIN_C5);
```

```
output_low (PIN_C6);
output_low (PIN_C7);
output_low (PIN_D7);
        break;
case 5:
output_low (PIN_C0);
output_low (PIN_C3);
output_high (PIN_C4);
output_low (PIN_C5);
output_low (PIN_C6);
output_low (PIN_C7);
output_high (PIN_D7);
        break;
case 6:
output_low (PIN_C0);
output_low (PIN_C3);
output_low (PIN_C4);
output_low (PIN_C5);
output_low (PIN_C6);
output_low (PIN_C7);
output_high (PIN_D7);
        break;
case 7:
output_low (PIN_C0);
output_high (PIN_C3);
```

```
        output_high (PIN_C4);
        output_high (PIN_C5);
        output_high (PIN_C6);
        output_low (PIN_C7);
        output_low (PIN_D7);
                break;
        case 8:
        output_low (PIN_C0);
        output_low (PIN_C3);
        output_low (PIN_C4);
        output_low (PIN_C5);
        output_low (PIN_C6);
        output_low (PIN_C7);
        output_low (PIN_D7);
                break;
        case 9:
        output_low (PIN_C0);
        output_low (PIN_C3);
        output_high (PIN_C4);
        output_low (PIN_C5);
        output_low (PIN_C6);
        output_low (PIN_C7);
        output_low (PIN_D7);
                break;
}
```

```
}

//////////////////////////////////////////////////////
  //              PID             //
//////////////////////////////////////////////////////
void PID ()
{
        unsigned long setpoint;                         //  Valor desejado
        signed long erro, derro;                          //  Erro e derivada do erro
        unsigned long real;
//      unsigned long       Kp_l, Kd_l, Ki_l;                   //  Valores lidos
        double   Kp, Kd, Ki;                //  Valores de Kp, Kd e Ki de 0 a 10
        double saida, pwm, real_conv;
                Kp=2.640;
                Kd=0.198;
                Ki=0.0088;
                set_adc_channel(4);
// Define o canal a ser utilizado para a próxima conversão A/D
                delay_us(10);
// Delay necessário para garantir o carregamento completo do capacitor interno
//do pino de amostragem
                setpoint = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
/*              set_adc_channel(1);
// Define o canal a ser utilizado para a próxima conversão A/D
```

```
                delay_us(10);
// Delay necessário para garantir o carregamento completo do capacitor interno
//do pino de amostragem
                Kp_l = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
                set_adc_channel(2);
// Define o canal a ser utilizado para a próxima conversão A/D
                delay_us(10);
// Delay necessário para garantir o carregamento completo do capacitor interno
//do pino de amostragem
                Kd_l = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
                set_adc_channel(3);
// Define o canal a ser utilizado para a próxima conversão A/D
                delay_us(10);
// Delay necessário para garantir o carregamento completo do capacitor interno
// do pino de amostragem
                Ki_l = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
*/
                set_adc_channel(0);
// Define o canal a ser utilizado para a próxima conversão A/D
                delay_us(10);
```

```
// Delay necessário para garantir o carregamento completo do capacitor interno
//do pino de amostragem
                real = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
        real_conv = ((double)real)/654.72;
        real_conv = real_conv*1023;
// Relaciona a maior tensão possível na entrada do PIC (3,2V) com o maior
//valor possível para a variável 'real'
                if(real_conv<=1023) {
                        real = (unsigned long)real_conv;
                }
                else {
                        real = 1023;
                }
/*              Kp = (double)Kp_l/1023;
                        // Tranforma Kp para um valor de 0 a 3
                Kp = Kp*3;
                Kd = (double)Kd_l/1023;
                        // Tranforma Kd para um valor de 0 a 3
                Kd = Kd*3;
                Ki = (double)Ki_l/1023;
                        // Tranforma Ki para um valor de 0 a 3
                Ki = Ki*3;
*/
                erro = setpoint - real;
                        // fórmula do erro
```

```
derro = (erro - erro_ant);
        // derivada do erro
soma_erro += (double)erro;
        // integral do erro
        erro_ant = erro;
        // guarda o erro atual
saida = Kp*(double)erro + Kd*(double)derro + Ki*soma_erro;
if (real<519)
{
        pwm=(0.027871836*saida)+9.80281474;
}
else if (real<699)
{
        pwm=(0.04469776*saida)+1.81182796;
}
else if (real<877)
{
        pwm=(0.10695253*saida)-42.25274725;
}
else
{
        pwm=(0.329912023*saida)-237.714286;
}
pwm=(pwm/100)*1023;
if(pwm<=1023) {
duty = (unsigned long)pwm;
```

```
                }
                else {
                duty=1023;   // Saturação do duty (não pode ser superior a 1023
                }
}
////////////////////////////////////////////////////
//      Função Principal     //
////////////////////////////////////////////////////
VOID MAIN()
{
   short int aberta_fechada;
        unsigned long digital;        // Variável para armazenar o valor digital lido
        unsigned int  uni, dez, cen;  // Variável para armazenar o valor digital lido
        float analogico;                    // Variável que recebe o valor analógico
        set_tris_a (0xff);
        set_tris_b (0x00);            // Configura as portas de entrada e saída
        set_tris_c (0x02);
        set_tris_d (0x00);
        uni=dez=cen=8;    // Inicializa o visor de 7 segmentos com o valor de 888
        unidade(uni);
        centena(cen);
        dezena(dez);
        aberta_fechada = input (pin_c1);
// Lê a chave e define controle de malha aberta ou fechada
```

```
   delay_ms (500);                          // Intervalo de inicialização do programa
        setup_adc_ports (ALL_ANALOG);       // Define as entradas analógicas
        setup_adc (adc_clock_internal);     // Define clock da conversão A/D
        setup_ccp1 (CCP_PWM);               // Configura RC2 como saída PWM
        setup_timer_2(T2_DIV_BY_16, 255, 1);
// Configura período do PWM: (1/clock)*4*t2div*(period+1)
        if (aberta_fechada==1)
//malha fechada com controlador
        {
                erro_ant = 0;
                soma_erro = 0;
                duty = 0;
                set_pwm1_duty(duty);        // Inicialmente configura PWM setado 0
                while (true)                // Loop de leitura e conversão A/D
                {
                        PID();
        // Leitura da entrada analógica em uma variável inteira utilizando
                        set_pwm1_duty (duty);       // Seta valor lido na saída PWM
                }
        }
        else                                        // malha aberta
        {
        set_adc_channel (4);
// Define o canal a ser utilizado para a próxima conversão A/D
                while (true)            // Loop de leitura e conversão A/D
```

```
                {
                duty = read_adc();
// Leitura da entrada analógica em uma variável inteira utilizando 10 bits (o a
//1023)
                set_pwm1_duty (duty);           // Seta valor lido na saída PWM


                //seta porcetagem no display
                analogico = (float)duty/1023;
                analogico = analogico*100;
                digital = (long)analogico;
// Converte o valor real em inteiro para mostrar no visor de 7 digitos


                cen = (digital/100);                    // Define a unidade
                dez = (digital-100*cen)/10;             // Define a dezena
                uni = (digital-100*cen-10*dez);         // Define a centena
                unidade(uni);
                centena(cen);
                dezena(dez);
                delay_ms (150);
// Intervalo entre as leituras analógicas e conversões A/D
                }
        }
}
```